PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F

A2

(11) International Publication Number: WO 98/19224

(43) International Publication Date: 7 May 1998 (07.05.98)

(21) International Application Number: PCT/US97/19391

(22) International Filing Date: 29 October 1997 (29.10.97)

(30) Priority Data:
08/741,862 29 October 1996 (29.10.96) US

(71) Applicant: OPEN MARKET, INC. [US/US]; 245 First Street, Cambridge, MA 02142 (US).

(72) Inventors: O'TOOLE, James, W., Jr.; 26 Concord Avenue No. 114, Cambridge, MA 02138 (US). GIFFORD, David, K.; 26 Pigeon Hill Road, Weston, MA 02693 (US).

(74) Agent: WALPERT, Gary, A.; Fish & Richardson PC, 225 Franklin Street, Boston, MA 02110 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: CONTROLLED TRANSFER OF INFORMATION IN COMPUTER NETWORKS

(57) Abstract

The present invention relates to techniques for controlling transfers of information in computer networks. One technique involves transmitting from a server computer to a client computer a document containing a channel object corresponding to a communication service, and storing an access ticket that indicates that a user of the client computer permits the information source computer to communicate with the user over a specified channel. Another technique involves transmitting smart digital offers based on information such as coupons and purchasing histories stored at the computer receiving the offer. Another technique involves transmitting from a server computer to a client computer a request for a user's personal profile information, and activating a client avatar that compares the request for personal profile information with a security profile of the user limiting access to personal profile information. Another technique involves transmitting from a server computer to a client computer a document containing an embedded link, activating the embedded link at the client computer and recording activation of the embedded link in a metering log.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

CONTROLLED TRANSFER OF INFORMATION IN COMPUTER NETWORKS

Reference to Appendix

Text Appendix A is being submitted with the present application.

Background of the Invention

The present invention relates to techniques for controlling transfers of information in computer networks, such as establishing communication channels between computers, transmitting smart digital offers based on information such as coupons and purchasing histories stored at the computer receiving the offer, automatically receiving data from a user's computer based on a personal profile and security profile of the user, and metering a user's access to linked information.

U.S. Patent Application Serial No. 08/168,519, filed December 16, 1993 by David K. Gifford and entitled "Digital Active Advertising," the entire disclosure of which is hereby incorporated herein in its entirety by reference, describes a network sales or payment system that includes at least a client computer and a payment computer. The client computer transmits a payment order and an authenticator to the payment computer. The payment computer verifies the authenticator, transmits a payment authorization message and an authenticator back to the client computer, and performs a payment settlement transaction.

U.S. Patent Application Serial No. 08/328,133, filed October 24, 1994 by Andrew C. Payne et al. and entitled "Network Sales System," the entire disclosure of which is hereby incorporated herein by reference, describes a network sales system in which a buyer computer transmits a payment order including a product identifier to a payment computer, which transmits an access message and an authenticator to a merchant

computer, which verifies the authenticator and causes the product to be sent to a user of the buyer computer. The payment computer stores the product identifier and the payment amount in a settlement database. A user at the buyer computer can transmit to the payment computer a request for an account statement, with an authenticator, and the payment computer verifies the authenticator and transmits a statement constructed from the settlement database to the buyer computer.

One known technique for transferring information in computer networks includes programming a computer to obtain packages of Web pages. The computer obtains the packages of Web pages automatically, on a periodic basis, without direct input from the user.

Summary of the Invention

One aspect of the invention features a network-based system for controlled transfer of information that includes a client computer, a server computer, and an information source computer interconnected by a computer network. The server computer transmits to the client computer a document containing a channel object corresponding to a communication service to be provided over an information transfer channel between the information source computer and the client computer. The client computer activates the channel object received from the server computer, and, in response to activation of the channel object, stores an access ticket that indicates that a user of the client computer permits the information source computer to communicate with the user over the channel. The information source computer transmits information to the client computer over the channel, and the client computer receives the information from the information source computer over the channel, based on the stored access ticket.

A user at the client computer can determine whether to activate a specific channel object received from the server computer and can specifically request that it be activated. Alternatively, the client computer can activate the channel object automatically if identifying data in the channel object specific to the information to be provided by the information source computers falls within parameters preset by the user such as a particular keyword phrase or a particular rating. The information transfer channel can be a broadcast or multicast channel, or it can simply be the computer network linking the client computer and the information source computer.

Another aspect of the invention features a network-based system for smart digital offer pricing that includes a client computer and an offer-providing server computer interconnected by a computer network. The offer-providing server computer transmits a document to the client computer that includes a smart digital offer object. The client computer stores user-specific information at the client computer, receives the document that includes the smart digital offer object, and activates the smart digital offer object at the client computer. Upon activation, the smart digital offer object provides an offer to the client computer based on the stored user-specific information. The client computer transmits an acceptance of the offer to the offer-providing server together with an authenticator. The offer-providing server verifies the authenticator and causes the offer to be fulfilled based on verification of the authenticator.

Because the smart digital offer object is executed at the client computer, it can efficiently use client-specific information that is stored at the client computer, even if the client computer is off-line and the

smart digital offer object has been received by e-mail, and it can minimize the load at the offer-providing server. In addition, the user-specific information examined by the smart digital offer object need not be revealed to the offer-providing server if the user does not accept the offer, because the client computer can contact the offer-providing server after activation of the smart digital offer object only if the user accepts the offer.

The user-specific information may be a coupon transmitted by a coupon-providing server computer to the client computer together with an authenticator. The client computer causes the coupon information and the authenticator to be stored, and the smart digital offer object, when it is activated, verifies the authenticator.

Another aspect of the invention features a network-based system for automatic transfer of information pertaining to a person profile of a user that includes a client computer and a server computer interconnected by a computer network. The server computer transmits to the client computer a document that includes a request for personal profile information pertaining to a user of the client computer. The client computer receives the document that includes the request for personal profile information, and activates a client avatar at the client computer. The client avatar compares the request for personal profile information with a security profile of the user limiting access to personal profile information and causes a subset of a personal profile of the user to be transmitted to the server computer based on the request for personal profile information and the security profile. The server computer transmits to the client computer information customized for the user based on the subset of the personal profile of the user.

The client avatar acts as an agent for the user by controlling the release of information from the client personal profile to the server computer. The client avatar makes it possible to store a single client personal profile at the client computer or an agency computer, rather than multiple personal profiles at multiple server computers, while at the same time limiting the release of certain information from the personal profile only to trusted servers or only upon specific authorization from the user.

Another aspect of the invention features a network-based system for metering of a user's access to linked information that includes a client computer and a server computer interconnected by a computer network. The server computer transmits to the client computer a document containing an embedded link. The client computer activates the embedded link when at least a portion of the document corresponding to the embedded link is displayed, records activation of the embedded link in a metering log, and causes information stored in the metering log pertaining to activation of the embedded link to be transmitted to the server computer.

This process makes it possible to charge a user on a per-usage basis for the user's access to information, without requiring the client computer to notify the server computer every time the user accesses the information. The per-usage charges can be assessed even if the client computer stores the documents in a cache from which the client computer periodically retrieves the documents. The information obtained from the metering log may alternatively be used solely for advertising feedback purposes, without any charges to the user.

Numerous other features, objects, and advantages of the invention will become apparent from the following

detailed description when read in connection with the accompanying drawings.

Brief Description of the Drawings

Fig. 1 is a block diagram of a network-based system for controlled transfer of information.

Fig. 2 is a flowchart diagram detailing the operation of the network-based system of Fig. 1.

Fig. 3 is a block diagram of a network-based system for smart digital offer pricing.

Figs. 4A and 4B are a flowchart diagram detailing the operation of the network-based system of Fig. 3.

Fig. 5 is a block diagram of a network-based system for transfer of information pertaining to a personal profile of a user.

Fig. 6 is a flowchart diagram detailing the operation of the network-based system of Fig. 5.

Fig. 7 is a block diagram of a network-based system for metering a user's access to linked information.

Fig. 8 is a flowchart diagram detailing the operation of the network-based system of Fig. 7.

Detailed Description

Referring to Fig. 1, a network-based system for controlled asynchronous transfer of information includes a client computer 10, operated by a user, that filters information transferred asynchronously to the client computer, a server computer 12 that transmits a document to the client computer containing a channel object that can be activated to authorize an asynchronous transfer of information, an information source computer 14 that asynchronously transfers the information, and an optional notification server 16 that acts as a trusted intermediary that filters asynchronously transferred information on behalf of the client computer. In certain implementations server computer 12 and information source

computer 14 are the same computer. As used herein, the term "asynchronous" transfer of information refers to a transfer of information from an information source computer that is initiated by the information source computer rather than by another computer to which the information source computer responds.

Client computer 10 or optional notification server 16 maintains an access control list 18 that stores access tickets that permit asynchronous transfers of information to the client computer or notification server. The access tickets are created upon activation of a channel object 20 received by client computer 10 from server computer 12. If optional notification server 16 is used to filter asynchronously transferred information on behalf of the client computer, the notification server maintains a list of messages 22 that can be retrieved by the client computer.

Referring to Fig. 2, in operation of the network-based system of Fig. 1, the client computer sends a message to the server computer (step 24) and the server responds by sending the client computer a document containing a channel object (step 26). Embedded within the channel object are a description of an asynchronous communication service, keywords describing the actual semantic content of the information to be transferred, an icon for identifying the asynchronous communication service to the user, a rating ("G," "PG," "R"), an identification of the size of the information block to be transferred, and any other information that might be useful t the user.

The description of the asynchronous communication service in the channel object may include a certificate that includes an identification of the supplier of the information to be transmitted to the client computer, as well as the supplier's public key, the certificate being

signed by a certifying authority. This public key will be used by the client computer to authenticate the information to be transmitted to the client computer by the information source computer.

The description of the asynchronous communication service in the channel object may specify a particular broadcast channel, such as a satellite feed channel on a portion of the internet or on a cable service, or a particular multicast channel, such as an Mbone channel. The description of the asynchronous communication service also specifies a particular time period during which the information will be transmitted asynchronously over the channel to many client computers.

When the document is displayed on the user computer, the icon contained in the channel object is displayed on the document as a representation of the channel object, and the user can determine from the document whether to authorize delivery of the content of the channel object as described in the document. The user can activate or select the channel object by clicking on a representation of the channel object on the document, or a channel object in a document or broadcast received by the client computer may be activated automatically by the computer if the keywords or the other identifying information contained in the channel object match preset parameters pre-programmed into the client computer as a personal profile of the user (step 28). For example, the user may pre-program the computer to search for a keyword phrase such as "BUGS BUNNY" to automatically activate channel objects pertaining to BUGS BUNNY. Similarly, the user may authorize automatic activation of channel objects containing an embedded "G" rating, or automatic activation of only one megabyte of information per week.

Activation of the channel object causes an access ticket containing the description of the asynchronous communication service to be added to the client control list in the client computer, or causes the access ticket to be sent to the notification server, which adds it to the access control list (step 30). The access ticket permits the information source computer to communicate asynchronously with the client computer over a channel specified by the channel object, which may be a broadcast or multicast channel at a specific time period, or which may be the computer network linking the client computer and the information source computer in the event that the information from the information source computer is to be received by means of an asynchronous communication over the computer network. Thus, the activation of a channel object initiates an asynchronous communication channel from the information source computer to the client computer and instructs the client computer that the information source computer is authorized to send information over the channel.

Once the channel object has been activated, the client computer notifies the server computer (or the information source computer, or another computer) that the access ticket was added to the access control list (step 32) and the server computer (or the information source computer, or another computer) records in a persistent database the client's interest in the channel object and sends a confirmation to the client computer that the client's interest in the channel object has been recorded (step 34).

The information source computer (which may have access to the persistent database mentioned above and therefore may be informed of the client's interest) asynchronously sends information to the client computer or the notification server (step 36) over the channel

specified by the channel object. The information includes an identification of its supplier and is signed using a private key of a public/private key pair. The client computer or the notification server accepts the information based on the presence of the appropriate access ticket in the access control list (step 38) corresponding to the supplier of the information and based on the client computer's use of the public key contained in the access ticket to ensure authenticity of the information.

For example, if the channel object and the access ticket specify a particular broadcast channel, or a particular multicast channel such as an Mbone channel, and specify a particular time period, the client computer will receive the information transmitted asynchronously by the information source computer to many client computers over the broadcast or multicast channel during that time period. The client computer filters the contents of the broadcast or multicast channel according to specifications derived from the access ticket. For example, the access ticket may specify that the information to be received by the client computer begins with a specific character or code that identifies the supplier of the information, its rating, or the content of the information. In addition, the access ticket may require the client computer to search for a specific keyword in the information, such as "BUGS BUNNY," before accepting the information.

Alternatively, if the channel object and the access ticket simply specify a particular supplier of information on the computer network, the client computer will receive information transmitted by the information source computer to the client computer over the computer network at any arbitrary time. The access ticket may specify a limit on the time during which the information

source computer is allowed to transmit information to the client computer. This time limit may originate from the channel object, and, in addition, the client computer may be programmed to allow the user to preset time limits on access tickets.

One specific implementation of an access control list is the use of a notification server that acts as a filtering mail gateway. The notification server, acting on behalf of the client computer, receives e-mail messages only from information source computers specified on the access control list. In other implementations the notification server is a file service operated by an internet service provider, or a part of the information systems department of a company that includes the client computer.

In another specific implementation the document containing the channel object that is transmitted by the server computer to the client computer specifies that the information from the information source computer will be encrypted, and that a key will be transmitted by the server computer to the user computer to decrypt the information upon the user paying a fee specified in the document. As an alternative, the user may be charged for use of the information from the information source computer according to the metering technique described below in connection with Figs. 7 and 8.

The client computer is programmed to permit the user to inquire which access tickets are in the user's access control list and to display the icons corresponding to each of the access tickets. These icons are included in the channel objects received by the client computer.

Channel objects may be embedded not only in documents or pages on the World Wide Web, but in an alternative implementation they may be embedded in e-mail

messages, OLE objects, ActiveX applets, etc. In fact, all of the communications between the server computer and the client computer and between the information source computer and the client computer may occur by e-mail, via compound documents, etc.

Referring to Fig. 3, another network-based system for controlled transfer of information includes a client computer 100, operated by a user, a coupon-providing server 102 that transmits a document to the client computer containing a coupon 104, and an offer-providing server 106 that transmits a document to the client computer containing or corresponding to a smart digital offer object 108 that calculates an offer based on the coupon 104 and on other information stored at the client computer. Offer-providing server 106 or optional intermediary server 111 may verify the information stored at the client computer on which the offer is based. The client computer 100 may store coupons 104 in coupon registry 110.

Referring to Figs. 4A and 4B, in operation of the network-based system of Fig. 3, the coupon-providing server sends a document to the client computer containing an embedded digital coupon (step 112). The coupon may be an executable program or program fragment expressed in machine-executable form, such as an ActiveX applet, and protected against unauthorized tampering by means of an authenticator such as a digital signature or MAC code (Message Authentication Code), or the coupon may be a digitally signed set of inputs to a program already residing at the client computer. The coupon contains a set of restrictions such as an expiration date, a product code or item number, and a discount amount. Alternatively, the coupon may simply contain a coded number that can be understood by the smart digital offer object described below.

The client computer retrieves the digital coupon from the document and stores it either in a coupon registry or separately (step 114). The client computer is programmed to periodically remind the user of the special rights or capabilities that possession of the coupon provides to the user, including the coupon's expiration date, using known methods such as pop-up windows and audiovisual prompts (step 116). The coupon may also contain a URL that is displayed to the user and on which a user can click to go to an offer-providing computer (a "store") that markets the product corresponding to the coupon as well as other products. Thus, the coupon acts as an advertising technique.

In one embodiment the coupon registry at the client computer is a purchasing history and the coupons are digital receipts identifying products purchased, dates of purchase, and possibly prices paid, together with authenticators of the digital receipts. The digital receipts function in the same manner as ordinary coupons because they will be used for the purpose of offering an adjusted price (typically a discounted price) to the user of the client computer. These digital receipts are transmitted from a server to the client computer together with authenticators upon completion of a purchase transaction.

The client computer fetches a document of web-based information from the offer-providing server that contains a smart digital offer object (step 118). The smart digital offer object may be an executable program or program fragment expressed in machine-executable form, such as an ActiveX applet, and protected against unauthorized tampering by means of an authenticator such as a digital signature or MAC code, or the smart digital offer object may be a digitally signed set of inputs to a program already residing at the client computer. The

smart digital offer object received by the client computer may be protected against unauthorized tampering by means of a digital signature or MAC code. In an alternative embodiment the smart digital offer object remains at the offer providing server and need not be protected against tampering. The client computer activates the smart digital offer object (step 120), and the smart digital offer object attempts to observe the parameters of the execution environment at the client machine, including the presence of coupons, and possibly other information such as a purchasing history recorded on the client computer.

If the smart digital offer object attempts to observe the purchasing history or certain other user-specific information, the client computer asks the user whether the user wishes to reveal the information (step 122). The user indicates whether release of the information is authorized (step 124), and the smart digital offer object then examines the coupon (including the coupon's authenticator), digital receipts (including authenticators) and other user-specific information authorized to be revealed by the user, and presents to the user an offer of a product or service (step 126). The execution environment at the client computer can under some circumstances change between steps 118 and 126. For example, the client computer may receive a coupon after step 118 occurs but before step 126 occurs. In one particular embodiment the client computer includes a client "avatar" of the type described below in connection with Figs. 5 and 6, which limits the release of certain information only to trusted servers, or only upon authorization from the client user, or both.

The terms or conditions of the offer, such as price and payment terms, are calculated by the smart digital offer object using formulas that depend on the

information contained in the digital coupons and the other information examined by the smart digital offer object, including the time of day, or user profile information such as membership codes, user's age, user's income, and other demographic information certified by an independent authority with an authenticator. When the user accepts the offer (step 128) the client computer sends a message to the offer-providing server indicating that the user has accepted the offer, or sends the message to an intermediary server that is trusted by the client computer to maintain the confidentiality of user-specific information and is trusted by the offer-providing server to verify the terms on which the offer was accepted (step 130). The message sent to the offer-providing server or the intermediary server includes the terms upon which the offer was accepted and also includes an authenticator. The offer-providing server or the intermediary server verifies the terms on which the offer was accepted by verifying the authenticator (step 132), and, if an intermediary server is used, the intermediary server reports the acceptance of the offer and the terms on which it was accepted to the offer-providing server. The offer-providing server then fulfills the offer by causing the offered product or service to be provided to the user (step 134).

The calculations of the terms and conditions of the offer may be performed in a smart card or other tamper-proof device on the client computer that is trusted by the offer-providing server. The smart card validates the smart digital offer object and the coupons and other signed information used by the smart digital offer object. If theses items are valid, the smart card calculates the terms and conditions of the offer based on the program fragments or parameters contained in the smart digital offer object, the coupon or coupons, and

the other information examined by the smart digital offer object. The smart card computes and signs a digest of the smart digital offer object, its inputs, and the terms and conditions calculated by the smart digital offer object. The client computer communicates this signed digest back to the offer-providing server with the acceptance message to be used as the authenticator. The acceptance message includes the terms and conditions of the offer. The smart card contains a secret key "K" that is used to create the signed digest. "K" is never released outside of the smart card. The smart card is designed to make it computationally infeasible to compute "K" even with possession of the device. The offer-providing server uses a signature checking key to check the authenticator.

Alternatively, the message sent by the client computer to the offer-providing server or the intermediary server indicating that the user has accepted the offer includes the smart digital offer object together with its authenticator, and it may also include the coupon and all other information examined by the smart digital offer object, together with authenticators (recall that coupons may include signatures). This enables the offer-providing server, or the intermediary server (which functions as an equivalent of a smart card on the client computer), to verify independently the authenticity of the smart digital offer object, as well as the authenticity of any information examined by the smart digital offer object that contains an authenticator such as a digital signature.

The coupon-providing server notifies the offer-providing server of the frequency of coupon distribution (step 136), and the offer-providing server notifies the coupon-providing server of the frequency of offer completion (step 138). This process makes it possible

for the coupon-providing and offer-providing servers to alter the terms of coupons and offers dynamically based on this information, possibly using complex control software.

Specific examples of security techniques (e.g., smart cards, signature verification) useful in connection with the smart digital offer technique described above are provided in the above-mentioned U.S. Patent Application Serial No. 08/168,519.

Specific examples of techniques for implementing objects such as the smart digital offer object and the coupons described above are described in Craig Brockschmidt, Inside OLE, second edition, Microsoft Press, 1995, and Adam Denning, OLE Controls Inside Out, Microsoft Press, 1995, the entire contents of which are hereby incorporated herein by reference.

An example of software code useful in implementing the smart digital offer pricing technique described above is attached hereto as Appendix A.

Referring to Fig. 5, another network-based system for controlled transfer of information includes a client computer 200, a server computer 202 and an optional agency computer 204. Client computer 200 or agency computer 204 stores a client personal profile 206 containing demographic data, current shopping interests and preferences, contact addresses, and other personal or semi-personal information. The client personal profile can include information that changes on a day-to-day basis, such as a purchasing history (which may be recorded in accordance with the techniques described in the above-mentioned U.S. Patent Application Serial No. 08/08/328,133), or a list of goods that the user wishes to buy (entered manually by the user in response to a prompt). Client computer 200 also stores a client security profile 208 that specifies that certain

information in client personal profile 206 should be disclosed to server computer 202 only to trusted servers or only upon authorization from the client user or both. A client "avatar" 210 located at client computer 200 acts as an agent for the user by controlling the release of information from client personal profile 206 to server computer 202.

Referring to Fig. 6, in operation of the network-based system of Fig. 5 the client computer obtains a document from the server computer that contains an offer/catalog description record (step 212) corresponding to an offer or catalog that will be sent to the client computer. The offer/catalog description record contains a profile query specifying the kinds of profile information that will be useful to the server computer in constructing a client-specific offer or in dynamically customizing the content of a catalog to be transmitted to the client computer. The offer/catalog description record also identifies the supplier of the record and the server computer to which the profile information should be sent, and contains the supplier's authenticating signature. Receipt of the offer/catalog description record by the client computer activates the client avatar (step 214). The client avatar compare the profile query in the offer/catalog description record with the security profile, which restricts the domain of profile information against which the profile query is processed (step 216).

If the profile query requests information that the security profile restricts only to trusted servers, then the client avatar determines whether the server computer is one of the trusted servers and, if so, checks the authenticating signature contained in the offer/catalog description record (step 217) (the client avatar may assume that if the supplier of the record is a trusted

supplier, then the server should be trusted too). If the profile query requests information that, according to the security profile, requires user authorization for release, then the client avatar prompts the user for authorization to release the information to the server computer (step 218) and the user indicates whether release of the information is authorized (step 220). Ordinarily, the user will not be prompted for authorization to release information to a trusted server, but the security profile can nevertheless be configured to require this for certain information.

After the client avatar determines which requested information can be released to the server computer, the client avatar transmits a subset of the client personal profile to the server computer, or sends an authorization message to the agency computer, which in turn transmits the subset of the client personal profile to the server computer (step 222). The subset includes all information in the client personal profile requested in the profile query and authorized for release to the server computer. Thus, the subset may not include all the information requested in the profile query. The server computer then transmits a client-specific sales offer or a customized document such as an electronic newspaper or magazine to the client computer based on the subset of the client personal profile received by the server computer (step 224), and the offer or document is displayed to the user at the client computer. The server computer may use the subset of the client personal profile to customize other web-based services offered to the user, including digital coupons, search services, and advertisements. Client-specific sales offers and coupons can be implemented in accordance with the smart digital offer technique described above in connection with Figs. 3 and 4A-4B. The server computer could alternatively use the subset of

the client personal profile to select or fabricate a channel object to send to the client computer, the channel object corresponding to a channel for asynchronous transfer of information to the client computer. The client computer can then activate the channel object in accordance with the technique described above in connection with Figs. 1 and 2. The server computer may even create a broadcast or multicast channel for the user by broadcasting or multicasting client-specific information and placing a specific identifying character or code at the beginning of the client-specific information. All of this can be accomplished using a single client personal profile stored at the client computer or agency computer, rather than multiple personal profiles stored at multiple server computers.

The security profile of the user can be developed progressively according to a scheme in which the security profile initially assumes that every supplier of offer/catalog description records is untrusted, every server is untrusted, and all information requires user authorization for release to every server. As profile queries are received by the client avatar, the client avatar queries the user whether the server computer should be trusted in the future (or whether the supplier of the offer/catalog description records should be trusted in the future, in which case the servers used by the trusted suppliers will be trusted too), and whether the requested information is authorized for release to untrusted servers. Based on the user's responses, the client avatar appropriately reconfigures the security profile.

In one embodiment, when the client avatar sends the subset of the client personal profile to the server computer, the client computer identifies the agency computer to the server computer. At the same time the

client avatar sends an authorization message to the agency computer authorizing release of certain information, or any and all information, from the client personal profile to the server computer. This allows the server computer to transmit profile queries to the agency computer and to receive from the agency computer subsets of the client personal profile, even when the client computer is off-line. The agency computer maintains an access control list corresponding to all of the authorization messages received from the client computer, so that the agency computer can know which information can be released to which servers.

Referring to Fig. 7, another network-based system for controlled transfer of information includes a client computer 300 that contains a metering log 302 for counting the number of times client computer 300 accesses certain information, a server computer 304 that provides documents to client computer 300, and an optional agency computer 306 that stores billing records 308 corresponding to the client computer's access to information.

Referring to Fig. 8, in operation of the network-based system of Fig. 7 the client computer first obtains valuable web-based information (step 310) in the form of a document containing an embedded active link that retrieves additional information and also implements a small program or applet. The active link may be embedded in the document by means of the known technique of ActiveX Controls. The client computer displays the document (step 312). When a user clicks on a representation of the active link (step 314) or, in an alternative embodiment described in detail below, when the active link is called by the browser at the client computer (step 316), the client computer activates the active link (step 318). Activation of the active link at

the client computer includes activation of the applet (step 320), which may fetch from the server computer, or elsewhere, a machine-executable program that is used for client-side metering of the end-user's access to valuable web-based information, as is explained below. The client computer may store the machine-executable program after it is first retrieved, so that subsequent activations of the applet do not require communication with another computer to obtain the program. Activation of the applet causes the client computer to record in the metering log the fact that a certain document, or a certain portion of the document, has been displayed (step 322).

The embedded active link may be a hyperlink that permits a user to navigate easily among documents by allowing the user to activate a hyperlink in a first document to obtain a second document, thereby making information contained in the documents readily accessible to the user. The retrieval of the second document can be implemented by the same applet that is used for the metering function. This can discourage disabling of or tampering with the metering function, especially if the embedded hyperlinks in a collection of documents are central to the utility of the collection of documents. In particular, the active hyperlink can check for the presence of a working metering log on the client computer before a second document is retrieved.

Other techniques for discouraging tampering could also be used. For example, the applet could fetch a program having a name that is changed on a frequent basis, where the scheme for changing the name is known only to the applet and where the applet is inoperable without the use of the program.

In certain embodiments the applet can use some or all of the techniques described above in connection with Figs. 3 and 4 to check for licenses, coupons,

subscription records, or access tickets in order to determine 1) whether to get a second document 2) which document to get, and/or 3) what information to record in the metering log.

As has been mentioned above, in certain embodiments the embedded active link is activated whenever it is called by a browser (step 316). In these embodiments the active link is a data record or tag record that automatically causes an embedded image to be retrieved and displayed at a certain location on the document. The applet is activated, and hence the metering function is activated, whenever the active link is initialized (i.e., whenever the document is displayed), or alternatively whenever the embedded image is displayed (i.e., whenever a certain portion of the document is displayed during a display refresh). The display of the embedded image can be implemented by the same applet that is used for the metering function, in order to discourage tampering with the metering function.

The embedded image may be transparent, in which case the sole practical function of the activation of the active link is to cause the client computer to activate the applet for metering of the user's access to information. The applet may record click activity on the transparent embedded image and then pass the click activity on to other objects in the document, thereby capturing detailed usage information that is stored in the metering log, such as the number and location of clicks. Because the active link is associated with an image (albeit a transparent image) the browser will not ignore it when the location of the transparent image is re-displayed.

In certain embodiments the applet described above is inoperable unless the active link that implements the applet includes a cryptographic validation signature.

This scheme ensures that the active links can be inserted into documents only by licensed authors.

The client computer periodically transmits the contents of the metering log to the server computer, or alternatively to the agency computer (step 324). If the contents of the metering log are transmitted to the agency computer, the agency computer enters the information contained in the metering log into detailed billing records, which may be records for a single client computer or many client computers, and the agency computer periodically transmits these billing records to the server computer. When the client computer accesses particularly valuable information the applet activated by the client computer may require the client computer to transmit the contents of the metering log immediately in order to prevent the client user from re-initializing the client computer and erasing its metering logs.

The information obtained from the metering log may be used solely for advertising feedback purposes, without any charges to the user. For example, the agency computer may be operated by an advertiser that is charged by the server computer on a per-usage basis whenever client computers display portions of documents on which advertisements are displayed. The client computer sends metering log information to the server computer and also to the agency computer so that the agency computer can know that the server computer has not tampered with the information.

There have been described novel and improved apparatus and techniques for controlled transfer of information in computer networks. It is evident that those skilled in the art may now make numerous uses and modifications of and departures from the specific embodiment described herein without departing from the inventive concept.

```
1   // CouponCtl.cpp : Implementation of the CCouponCtrl OLE control class.
2   #include <windows.h>
3
4   #include "stdafx.h"
5   #include "coupon.h"
6   #include "CouponCtl.h"
7   #include "CouponPpg.h"
8
9   #include <winreg.h>
10
11  #ifdef _DEBUG
12  #define new DEBUG_NEW
13  #undef THIS_FILE
14  static char THIS_FILE[] = __FILE__;
15  #endif
16
17
18  static char *radix64encode_noslash(char *in, int len);
19  static char *radix64decode_noslash(char *in, int len, int *output_len);
20  static char *common_radix64encode(unsigned char *table, char *in, int len);
21  static char *radix64encode_noslash(char *in, int len);
22  static char *commonradix64decode(unsigned char *rev_table, char *in,
23                                   int len, int *output_len);
24  static char *radix64decode_noslash(char *in, int len, int *output_len);
25
26  static int isCreated = 0;
27
28  IMPLEMENT_DYNCREATE(CCouponCtrl, COleControl)
29
30
31  /////////////////////////////////////////////////////////////////////////////
32  // Message map
33
34  BEGIN_MESSAGE_MAP(CCouponCtrl, COleControl)
35          //{{AFX_MSG_MAP(CCouponCtrl)
36          ON_WM_LBUTTONDBLCLK()
37          //}}AFX_MSG_MAP
38          ON_OLEVERB(AFX_IDS_VERB_EDIT, OnEdit)
39          ON_OLEVERB(AFX_IDS_VERB_PROPERTIES, OnProperties)
40  END_MESSAGE_MAP()
41
42
43  /////////////////////////////////////////////////////////////////////////////
44  // Dispatch map
45
46  BEGIN_DISPATCH_MAP(CCouponCtrl, COleControl)
47          //{{AFX_DISPATCH_MAP(CCouponCtrl)
48          DISP_PROPERTY_EX(CCouponCtrl, "UniqueID", GetUniqueID, SetUniqueID, VT
    +_BSTR)
49          DISP_PROPERTY_EX(CCouponCtrl, "StoreID", GetStoreID, SetStoreID, VT_BS
    +TR)
50          DISP_PROPERTY_EX(CCouponCtrl, "DiscountRate", GetDiscountRate, SetDisc
    +ountRate, VT_R8)
51          DISP_PROPERTY_EX(CCouponCtrl, "DiscountAmount", GetDiscountAmount, Set
    +DiscountAmount, VT_R8)
52          //}}AFX_DISPATCH_MAP
53          DISP_FUNCTION_ID(CCouponCtrl, "AboutBox", DISPID_ABOUTBOX, AboutBox, V
    +T_EMPTY, VTS_NONE)
54  END_DISPATCH_MAP()
55
56
57  /////////////////////////////////////////////////////////////////////////////
58  // Event map
59
60  BEGIN_EVENT_MAP(CCouponCtrl, COleControl)
61          //{{AFX_EVENT_MAP(CCouponCtrl)
62          // NOTE - ClassWizard will add and remove event map entries
63          //    DO NOT EDIT what you see in these blocks of generated code !
64          //}}AFX_EVENT_MAP
65  END_EVENT_MAP()
```



```
66
67
68  /////////////////////////////////////////////////////////////////////////////
69  // Property pages
70
71  // TODO: Add more property pages as needed.  Remember to increase the count!
72  BEGIN_PROPPAGEIDS(CCouponCtrl, 1)
73          PROPPAGEID(CCouponPropPage::guid)
74  END_PROPPAGEIDS(CCouponCtrl)
75
76
77  /////////////////////////////////////////////////////////////////////////////
78  // Initialize class factory and guid
79
80  IMPLEMENT_OLECREATE_EX(CCouponCtrl, "COUPON.CouponCtrl.1",
81          0xebf68c63, 0x234a, 0x11d0, 0xa0, 0x21, 0x44, 0x45, 0x53, 0x54, 0, 0)
82
83
84  /////////////////////////////////////////////////////////////////////////////
85  // Type library ID and version
86
87  IMPLEMENT_OLETYPELIB(CCouponCtrl, _tlid, _wVerMajor, _wVerMinor)
88
89
90  /////////////////////////////////////////////////////////////////////////////
91  // Interface IDs
92
93  const IID BASED_CODE IID_DCoupon =
94                  { 0xebf68c61, 0x234a, 0x11d0, { 0xa0, 0x21, 0x44, 0x45, 0x53,
    +0x54, 0, 0 } };
95  const IID BASED_CODE IID_DCouponEvents =
96                  { 0xebf68c62, 0x234a, 0x11d0, { 0xa0, 0x21, 0x44, 0x45, 0x53,
    +0x54, 0, 0 } };
97
98
99  /////////////////////////////////////////////////////////////////////////////
100 // Control type information
101
102 static const DWORD BASED_CODE _dwCouponOleMisc =
103         OLEMISC_ACTIVATEWHENVISIBLE |
104         OLEMISC_SETCLIENTSITEFIRST |
105         OLEMISC_INSIDEOUT |
106         OLEMISC_CANTLINKINSIDE |
107         OLEMISC_RECOMPOSEONRESIZE;
108
109 IMPLEMENT_OLECTLTYPE(CCouponCtrl, IDS_COUPON, _dwCouponOleMisc)
110
111 void showError(LONG rc, char * msg);
112
113 void showError(LONG rc, char * msg)
114 {
115         char tmpBuf[100];
116
117     if (rc != ERROR_SUCCESS)
118         {
119                 sprintf(tmpBuf, "Error %s : %d", msg, rc);
120
121                 MessageBoxEx(NULL, tmpBuf, "Digital Coupon",
122                         MB_OK, MAKELANGID(LANG_ENGLISH, SUBLANG_ENGLISH_US));
123     }
124 }
125
126 /////////////////////////////////////////////////////////////////////////////
127 // CCouponCtrl::CCouponCtrlFactory::UpdateRegistry -
128 // Adds or removes system registry entries for CCouponCtrl
129
130 BOOL CCouponCtrl::CCouponCtrlFactory::UpdateRegistry(BOOL bRegister)
131 {
132         // TODO: Verify that your control follows apartment-model threading ru
    +les.
```

Oct 29 1996 16:06:19 CouponCtl.cpp Page 3

```
133         // Refer to MFC TechNote 64 for more information.
134         // If your control does not conform to the apartment-model rules, then
135         // you must modify the code below, changing the 6th parameter from
136         // afxRegInsertable | afxRegApartmentThreading to afxRegInsertable.
137
138         if (bRegister)
139                 return AfxOleRegisterControlClass(
140                         AfxGetInstanceHandle(),
141                         m_clsid,
142                         m_lpszProgID,
143                         IDS_COUPON,
144                         IDB_COUPON,
145                         afxRegInsertable | afxRegApartmentThreading,
146                         _dwCouponOleMisc,
147                         _tlid,
148                         _wVerMajor,
149                         _wVerMinor);
150         else
151                 return AfxOleUnregisterClass(m_clsid, m_lpszProgID);
152 }
153
154
155 /////////////////////////////////////////////////////////////////////////////
156 // Licensing strings
157
158 static const TCHAR BASED_CODE _szLicFileName[] = _T("coupon.lic");
159
160 static const WCHAR BASED_CODE _szLicString[] =
161         L"Copyright (c) 1996 ";
162
163
164 /////////////////////////////////////////////////////////////////////////////
165 // CCouponCtrl::CCouponCtrlFactory::VerifyUserLicense -
166 // Checks for existence of a user license
167
168 BOOL CCouponCtrl::CCouponCtrlFactory::VerifyUserLicense()
169 {
170         return AfxVerifyLicFile(AfxGetInstanceHandle(), _szLicFileName,
171                 _szLicString);
172 }
173
174
175 /////////////////////////////////////////////////////////////////////////////
176 // CCouponCtrl::CCouponCtrlFactory::GetLicenseKey -
177 // Returns a runtime licensing key
178
179 BOOL CCouponCtrl::CCouponCtrlFactory::GetLicenseKey(DWORD dwReserved,
180         BSTR FAR* pbstrKey)
181 {
182         if (pbstrKey == NULL)
183                 return FALSE;
184
185         *pbstrKey = SysAllocString(_szLicString);
186         return (*pbstrKey != NULL);
187 }
188
189
190 /////////////////////////////////////////////////////////////////////////////
191 // CCouponCtrl::CCouponCtrl - Constructor
192
193 CCouponCtrl::CCouponCtrl()
194 {
195         OSL_KeyStruct *keyStruct;
196
197         char keyfile[] = "c:\\oml\\slsdkpro\\conf\\demokey.kf";
198     char keyfile2[] = "c:\\oml\\slsdkpro\\conf\\secret.kf";
199
200         InitializeIIDs(&IID_DCoupon, &IID_DCouponEvents);
201
202         // TODO: Initialize your control's instance data here.
```

Oct 29 1996 16:06:19 CouponCtl.cpp Page 4

```
203         //m_StoreID = _T ("110000");
204         m_StoreID = _T ("1108");
205
206         m_UniqueID = _T ("");
207         m_DiscountRate = 0.0;
208         m_DiscountAmount = 0.0;
209
210         if ( OSL_LoadKeyCacheFromFile( &m_keyCache, &m_err, keyfile2) )
211         {
212                 TRACE(m_err.message);
213                 return ;
214         }
215
216
217         if ( OSL_GetKeyFromCache( &m_key, &m_err, m_StoreID,
218                         m_keyCache) )
219         {
220                 TRACE(m_err.message);
221                 return ;
222         }
223
224         keyStruct = (OSL_KeyStruct *) m_key;
225
226         strcpy(m_skey,keyStruct->key);
227     strcpy(m_kid, keyStruct->kid);
228
229 }
230
231
232 /////////////////////////////////////////////////////////////////////////////
233 // CCouponCtrl::~CCouponCtrl - Destructor
234
235 CCouponCtrl::~CCouponCtrl()
236 {
237         // TODO: Cleanup your control's instance data here.
238 }
239
240
241 /////////////////////////////////////////////////////////////////////////////
242 // CCouponCtrl::OnDraw - Drawing function
243
244 void CCouponCtrl::OnDraw(
245                         CDC* pdc, const CRect& rcBounds, const CRect& rcInvali
    d)
246 {
247         RECT x = rcBounds;
248         CString showText;
249         char buf[10];
250
251         //pdc->FillRect(rcBounds, CBrush::FromHandle((HBRUSH)GetStockObject(WH
    ITE_BRUSH)));
252         pdc->FillRect(rcBounds, CBrush::FromHandle((HBRUSH)GetStockObject(GRAY
    _BRUSH)));
253         //pdc->Draw3dRect( &x, 0x0f0f0f0f, 0xf0f0f0f0 );
254         pdc->Draw3dRect( &x, RGB(255,255,0), 0xf0f0f0f0 );
255
256         showText = "";
257         sprintf(buf, "%2d", (long) (m_DiscountRate * 100.0) );
258         showText += buf;
259         showText += "% OFF ";
260         showText += m_UniqueID;
261
262         pdc->DrawText ( showText,  &x, DT_SINGLELINE |DT_CENTER |DT_VCENTER);
263
264 }
265
266
267 /////////////////////////////////////////////////////////////////////////////
268 // CCouponCtrl::DoPropExchange - Persistence support
269
```

```
270 void CCouponCtrl::DoPropExchange(CPropExchange* pPX)
271 {
272         OM_aa options;
273         char payload[1000];
274         LPCTSTR orgBuf;
275         char discount[20];
276
277         ExchangeVersion(pPX, MAKELONG(_wVerMinor, _wVerMajor));
278         COleControl::DoPropExchange(pPX);
279
280         // TODO: Call PX_ functions for each persistent custom property.
281 //      PX_String (pPX, _T ("StoreID"), m_StoreID, _T ("") );
282         PX_String (pPX, _T ("UniqueID"), m_UniqueID, _T ("") );
283         PX_Double (pPX, _T ("DiscountAmount"), m_DiscountAmount, 0.0 );
284         PX_Double (pPX, _T ("DiscountRate"), m_DiscountRate, 0.0 );
285
286
287         if ( !pPX -> IsLoading () )
288         {
289                 options = OM_aaCreate();
290
291                 orgBuf = m_UniqueID;
292                 OM_aaAddEntry(options, "UniqueID", (void *) orgBuf);
293                 sprintf(discount, "%4.2lf", m_DiscountRate);
294                 OM_aaAddEntry(options, "DiscountRate", discount);
295
296                 if ( OSL_mkpayload(m_kid, m_skey, "env", options, payload,
297                         sizeof(payload), &m_err) ) {
298                         m_Ticket = _T("");
299
300                 }
301                 else
302                 {
303 //                      encBuf = radix64encode_noslash((char *) payload, strle
    n(payload));
304 //                      m_Ticket = encBuf;
305 //                      free( (void *) encBuf);
306                         m_Ticket = payload;
307                         isCreated = 1;
308
309                 }
310         }
311
312         PX_String (pPX, _T ("Ticket"), m_Ticket, _T ("") );
313
314 }
315
316
317 /////////////////////////////////////////////////////////////////////////////
318 // CCouponCtrl::OnResetState - Reset control to default state
319
320 void CCouponCtrl::OnResetState()
321 {
322         COleControl::OnResetState();  // Resets defaults found in DoPropExchan
    ge
323
324         // TODO: Reset any other control state here.
325 }
326
327
328 /////////////////////////////////////////////////////////////////////////////
329 // CCouponCtrl::AboutBox - Display an "About" box to the user
330
331 void CCouponCtrl::AboutBox()
332 {
333         CDialog dlgAbout(IDD_ABOUTBOX_COUPON);
334         dlgAbout.DoModal();
335 }
336
337
```



```
338 /////////////////////////////////////////////////////////////////////////////
339 // CCouponCtrl message handlers
340
341 BSTR CCouponCtrl::GetUniqueID()
342 {
343         return m_UniqueID.AllocSysString();
344 }
345
346 void CCouponCtrl::SetUniqueID(LPCTSTR lpszNewValue)
347 {
348         m_UniqueID = lpszNewValue;
349         SetModifiedFlag();
350 }
351
352 BSTR CCouponCtrl::GetStoreID()
353 {
354         return m_StoreID.AllocSysString();
355 }
356
357 void CCouponCtrl::SetStoreID(LPCTSTR lpszNewValue)
358 {
359         m_StoreID = lpszNewValue;
360         SetModifiedFlag();
361 }
362
363 double CCouponCtrl::GetDiscountRate()
364 {
365         return m_DiscountRate;
366 }
367
368 void CCouponCtrl::SetDiscountRate(double newValue)
369 {
370         m_DiscountRate = newValue;
371         SetModifiedFlag();
372 }
373
374
375
376 double CCouponCtrl::GetDiscountAmount()
377 {
378         return m_DiscountAmount;
379 }
380
381 void CCouponCtrl::SetDiscountAmount(double newValue)
382 {
383
384         m_DiscountAmount = newValue;
385
386         SetModifiedFlag();
387 }
388
389 void CCouponCtrl::OnLButtonDblClk(UINT nFlags, CPoint point)
390 {
391         // TODO: Add your message handler code here and/or call default
392         LPCTSTR orgBuf;
393         CString couponKey;
394         LONG rc;
395         int res;
396
397         HKEY hkey;
398         DWORD disposition;
399         char rate[10];
400         char msg[200];
401
402
403         sprintf(msg, "You are picking up a digital coupon which offers %d perc
    ent off %s of store %s",
404                 (int)(m_DiscountRate * 100.0), m_UniqueID, m_StoreID);
405
406         res = MessageBoxEx(NULL, msg, "Digital Coupon",
```


WO 98/19224 -27- PCT/US97/19391

Oct 29 1996 16:06:19 CouponCtl.cpp Page 7

```
407                     MB_ICONQUESTION|MB_OKCANCEL, MAKELANGID(LANG_ENGLISH,
    SUBLANG_ENGLISH_US));
408
409         if (res == IDCANCEL) return;
410
411         couponKey = _T ("");
412         couponKey += "Digital Coupons\\";
413         couponKey += m_StoreID;
414         couponKey += "\\";
415         couponKey += m_UniqueID;
416
417
418     rc  = RegCreateKeyEx(
419                 HKEY_CURRENT_USER,
420                 couponKey,
421                 0,
422                 "Digital-Coupons",      // address of class string
423                 REG_OPTION_NON_VOLATILE,        // special options flag
424                 KEY_ALL_ACCESS, // desired security access
425                 NULL,   // address of key security structure
426                 &hkey,  // address of buffer for opened handle
427                 &disposition    // address of disposition value buffer
428     );
429
430         showError(rc, "Creating Keys");
431
432         sprintf (rate, "%4.2lf", m_DiscountRate);
433
434         rc = RegSetValueEx(
435     hkey,       // handle of key to set value for
436     "Discount Rate",    // address of value to set
437     0,  // reserved
438     REG_SZ,     // flag for value type
439     (CONST BYTE *) rate,        // address of value data
440     strlen (rate)       // size of value data
441     );
442
443         showError(rc, "Setting Discount Rate");
444
445                 orgBuf = m_Ticket;
446
447         rc = RegSetValueEx(hkey,"Ticket",0, REG_SZ,
448                 (CONST BYTE *) orgBuf, m_Ticket.GetLength());
449
450     showError(rc, "Setting Ticket");
451
452
453 //      COleControl::OnLButtonDblClk(nFlags, point);
454 }
455
456
457 /* Radix-64 encoding and decoding routines.
458  * See RFC1421 for details.
459  */
460
461
462 /* This is a modified version of RADIX64, the "normal" one, has been   */
463 /* modified to replace the / and + with the * and @ chars.             */
464 /* The "modified" version is called the _noslash version's. These work */
465 /* better in URL's.                                                    */
466
467
468 /* encode tables */
469
470 static unsigned char table_noslash[64] = {
471     'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H',
472     'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P',
473     'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X',
474     'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f',
475     'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n',
```

Oct 29 1996 16:06:19 CouponCtl.cpp Page 8

```
476     'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
477     'w', 'x', 'y', 'z', '0', '1', '2', '3',
478 '4', '5', '6', '7', '8', '9', '@', '*'};
479
480
481
482
483 /* decode tables */
484
485
486 static unsigned char rev_table_noslash[256] = {
487     255, 255, 255, 255, 255, 255, 255, 255,
488     255, 255, 255, 255, 255, 255, 255, 255,
489     255, 255, 255, 255, 255, 255, 255, 255,
490     255, 255, 255, 255, 255, 255, 255, 255,
491     255, 255, 255, 255, 255, 255, 255, 255,
492     255, 255, 63, 255, 255, 255, 255, 255,
493     52, 53, 54, 55, 56, 57, 58, 59,
494     60, 61, 255, 255, 255, 255, 255, 255,
495     62, 0, 1, 2, 3, 4, 5, 6,
496     7, 8, 9, 10, 11, 12, 13, 14,
497     15, 16, 17, 18, 19, 20, 21, 22,
498     23, 24, 25, 255, 255, 255, 255, 255,
499     255, 26, 27, 28, 29, 30, 31, 32,
500     33, 34, 35, 36, 37, 38, 39, 40,
501     41, 42, 43, 44, 45, 46, 47, 48,
502     49, 50, 51, 255, 255, 255, 255, 255,
503     255, 255, 255, 255, 255, 255, 255, 255,
504     255, 255, 255, 255, 255, 255, 255, 255,
505     255, 255, 255, 255, 255, 255, 255, 255,
506     255, 255, 255, 255, 255, 255, 255, 255,
507     255, 255, 255, 255, 255, 255, 255, 255,
508     255, 255, 255, 255, 255, 255, 255, 255,
509     255, 255, 255, 255, 255, 255, 255, 255,
510     255, 255, 255, 255, 255, 255, 255, 255,
511     255, 255, 255, 255, 255, 255, 255, 255,
512     255, 255, 255, 255, 255, 255, 255, 255,
513     255, 255, 255, 255, 255, 255, 255, 255,
514     255, 255, 255, 255, 255, 255, 255, 255,
515     255, 255, 255, 255, 255, 255, 255, 255,
516     255, 255, 255, 255, 255, 255, 255, 255,
517     255, 255, 255, 255, 255, 255, 255, 255,
518     255, 255, 255, 255, 255, 255, 255, 255};
519
520
521
522
523
524
525 #define sextet1(p)      ((((p)[0]) >> 2) & 0x3f)
526 #define sextet2(p)      ((((p)[0]) & 0x3) << 4) | ((((p)[1]) & 0xf0) >> 4))
527 #define sextet3(p)      ((((p)[1]) & 0xf) << 2) | ((((p)[2]) >> 6) & 0x3))
528 #define sextet4(p)      ((((p)[2]) & 0x3f)
529
530
531 static char *radix64encode_noslash(char *in, int len)
532 {
533     return (common_radix64encode(table_noslash, in, len));
534 }
535
536
537
538
539 static char *common_radix64encode(unsigned char *table, char *in, int len)
540 {
541     int i;
542     char *buf, *p;
543     int buflen;
544
545     /* Check parameters. */
```

```
546     if (in == 0 || len == 0) {
547         return (NULL);
548     }
549     buflen = ((len - 1) / 3 + 1) * 4;
550     if ((buf = (char *) malloc(buflen + 1)) == NULL) {
551         return (NULL);
552     }
553     /*
554      * Encode all but the last 1-3 bytes, since the result may have to to be
555      * padded.
556      */
557
558     p = buf;
559     for (i = 0; i < len - 3; i += 3) {
560         *p++ = table[sextet1(&in[i])];
561         *p++ = table[sextet2(&in[i])];
562         *p++ = table[sextet3(&in[i])];
563         *p++ = table[sextet4(&in[i])];
564     }
565
566     /* Encode remaining bytes. */
567     switch (len - i) {
568         case 1:
569             *p++ = table[sextet1(&in[i])];
570             *p++ = table[sextet2(&in[i])];
571             *p++ = '=';
572             *p++ = '=';
573             break;
574         case 2:
575             *p++ = table[sextet1(&in[i])];
576             *p++ = table[sextet2(&in[i])];
577             *p++ = table[sextet3(&in[i])];
578             *p++ = '=';
579             break;
580         case 3:
581             *p++ = table[sextet1(&in[i])];
582             *p++ = table[sextet2(&in[i])];
583             *p++ = table[sextet3(&in[i])];
584             *p++ = table[sextet4(&in[i])];
585             break;
586         default:
587             return (NULL);
588     }
589     *p = 0;
590     return (buf);
591 }
592
593
594
595 /* Decode radix-64 into binary. */
596
597
598
599
600
601 #define octet1(p)       (((p)[0] << 2) | (((p)[1] >> 4) & 0x3))
602 #define octet2(p)       ((((p)[1] << 4) & 0xf0) | (((p)[2] >> 2) & 0xf))
603 #define octet3(p)       (((((p)[2]) & 0x3) << 6) | ((p)[3]))
604
605
606 static char *radix64decode_noslash(char *in, int len, int *output_len)
607 {
608     return (commonradix64decode(rev_table_noslash, in, len, output_len));
609 }
610
611
612 static char *commonradix64decode(unsigned char *rev_table, char *in, int len,
    int *output_len)
613 {
614     int i;
```



```
615     unsigned char datum[4];
616     unsigned char *buf, *p;
617     int buflen;
618
619     if (in == NULL || len == 0) {
620 /*      fprintf(stderr, "decode: bad parameters %x %d\n", 0, len); */
621         return (NULL);
622     }
623     /*
624      * By definition, the length of the input buffer must be a multiple of 4.
625      */
626
627     if (len % 4 != 0) {
628 /*      fprintf(stderr, "decode: input length not a multiple of 4\n"); */
629         return (NULL);
630     }
631     buflen = (len * 3) / 4;
632
633     /* Trim padding. */
634     if (in[len - 1] == '=')
635         buflen--;
636     if (in[len - 2] == '=')
637         buflen--;
638
639     if ((buf = (unsigned char *) malloc(buflen + 1)) == NULL) {
640 /*      fprintf(stderr, "decode: unable to allocate %d bytes\n", buflen); */
641         return (NULL);
642     }
643     /* Decode all but the last four bytes. */
644
645     p = buf;
646     for (i = 0; i < len - 4; i += 4) {
647         datum[0] = rev_table[in[i]];
648         datum[1] = rev_table[in[i + 1]];
649         datum[2] = rev_table[in[i + 2]];
650         datum[3] = rev_table[in[i + 3]];
651         *p++ = octet1(datum);
652         *p++ = octet2(datum);
653         *p++ = octet3(datum);
654     }
655
656     /* And the last four bytes... */
657
658     datum[0] = rev_table[in[i]];
659     datum[1] = rev_table[in[i + 1]];
660     datum[2] = rev_table[in[i + 2]];
661     datum[3] = rev_table[in[i + 3]];
662     *p++ = octet1(datum);
663     if (in[i + 2] != '=') {
664         *p++ = octet2(datum);
665         if (in[i + 3] != '=') {
666             *p++ = octet3(datum);
667         }
668     }
669     *output_len = buflen;
670         * (buf + buflen ) = 0;
671     return ((char *) buf);
672 }
673
```

Oct 29 1996 16:07:15 CouponCtl.h Page 1

```
1   // CouponCtl.h : Declaration of the CCouponCtrl OLE control class.
2
3   ////////////////////////////////////////////////////////////////////////////
4   // CCouponCtrl : See CouponCtl.cpp for implementation.
5   extern "C" {
6
7   #include "doint.h"
8
9   }
10
11
12  class CCouponCtrl : public COleControl
13  {
14          DECLARE_DYNCREATE(CCouponCtrl)
15
16  // Constructor
17  public:
18          CCouponCtrl();
19
20  // Overrides
21
22          // Drawing function
23          virtual void OnDraw(
24                                  CDC* pdc, const CRect& rcBounds, const CRect&
    rcInvalid);
25          // Persistence
26          virtual void DoPropExchange(CPropExchange* pPX);
27
28          // Reset control state
29          virtual void OnResetState();
30
31  // Implementation
32  protected:
33          ~CCouponCtrl();
34
35          BEGIN_OLEFACTORY(CCouponCtrl)        // Class factory and guid
36                  virtual BOOL VerifyUserLicense();
37                  virtual BOOL GetLicenseKey(DWORD, BSTR FAR*);
38          END_OLEFACTORY(CCouponCtrl)
39
40          DECLARE_OLETYPELIB(CCouponCtrl)      // GetTypeInfo
41          DECLARE_PROPPAGEIDS(CCouponCtrl)     // Property page IDs
42          DECLARE_OLECTLTYPE(CCouponCtrl)         // Type name and misc status
43
44  // Message maps
45          //{{AFX_MSG(CCouponCtrl)
46          afx_msg void OnLButtonDblClk(UINT nFlags, CPoint point);
47          //}}AFX_MSG
48          DECLARE_MESSAGE_MAP()
49
50  // Dispatch maps
51          //{{AFX_DISPATCH(CCouponCtrl)
52          afx_msg BSTR GetUniqueID();
53          afx_msg void SetUniqueID(LPCTSTR lpszNewValue);
54          afx_msg BSTR GetStoreID();
55          afx_msg void SetStoreID(LPCTSTR lpszNewValue);
56          afx_msg double GetDiscountRate();
57          afx_msg void SetDiscountRate(double newValue);
58          afx_msg double GetDiscountAmount();
59          afx_msg void SetDiscountAmount(double newValue);
60          //}}AFX_DISPATCH
61          DECLARE_DISPATCH_MAP()
62
63          afx_msg void AboutBox();
64
65  // Event maps
66          //{{AFX_EVENT(CCouponCtrl)
67          //}}AFX_EVENT
68          DECLARE_EVENT_MAP()
69
```

Oct 29 1996 16:07:15 CouponCtl.h Page 2

```
70
71  // Dispatch and event IDs
72  public:
73          enum {
74          //{{AFX_DISP_ID(CCouponCtrl)
75          dispidUniqueID = 1L,
76          dispidStoreID = 2L,
77          dispidDiscountRate = 3L,
78          dispidDiscountAmount = 4L,
79          //}}AFX_DISP_ID
80          };
81  private:
82          CString m_StoreID;
83          CString m_UniqueID;
84          double m_DiscountRate;
85          double m_DiscountAmount;
86          CString m_Ticket;
87
88          OSL_Error m_err;
89          OSL_Key   m_key;
90          OSL_KeyCache m_keyCache;
91          char m_skey[100];
92          char m_kid[20];
93
94  };
```

Oct 29 1996 16:06:20 CouponPpg.cpp Page 1

```
1   // CouponPpg.cpp : Implementation of the CCouponPropPage property page class.
2
3   #include "stdafx.h"
4   #include "coupon.h"
5   #include "CouponPpg.h"
6
7   #ifdef _DEBUG
8   #define new DEBUG_NEW
9   #undef THIS_FILE
10  static char THIS_FILE[] = __FILE__;
11  #endif
12
13
14  IMPLEMENT_DYNCREATE(CCouponPropPage, COlePropertyPage)
15
16
17  ////////////////////////////////////////////////////////////////////////////
18  // Message map
19
20  BEGIN_MESSAGE_MAP(CCouponPropPage, COlePropertyPage)
21          //{{AFX_MSG_MAP(CCouponPropPage)
22          // NOTE - ClassWizard will add and remove message map entries
23          //    DO NOT EDIT what you see in these blocks of generated code !
24          //}}AFX_MSG_MAP
25  END_MESSAGE_MAP()
26
27
28  ////////////////////////////////////////////////////////////////////////////
29  // Initialize class factory and guid
30
31  IMPLEMENT_OLECREATE_EX(CCouponPropPage, "COUPON.CouponPropPage.1",
32          0xebf68c64, 0x234a, 0x11d0, 0xa0, 0x21, 0x44, 0x45, 0x53, 0x54, 0, 0)
33
34
35  ////////////////////////////////////////////////////////////////////////////
36  // CCouponPropPage::CCouponPropPageFactory::UpdateRegistry -
37  // Adds or removes system registry entries for CCouponPropPage
38
39  BOOL CCouponPropPage::CCouponPropPageFactory::UpdateRegistry(BOOL bRegister)
40  {
41          if (bRegister)
42                  return AfxOleRegisterPropertyPageClass(AfxGetInstanceHandle(),
43                          m_clsid, IDS_COUPON_PPG);
44          else
45                  return AfxOleUnregisterClass(m_clsid, NULL);
46  }
47
48
49  ////////////////////////////////////////////////////////////////////////////
50  // CCouponPropPage::CCouponPropPage - Constructor
51
52  CCouponPropPage::CCouponPropPage() :
53          COlePropertyPage(IDD, IDS_COUPON_PPG_CAPTION)
54  {
55          //{{AFX_DATA_INIT(CCouponPropPage)
56          m_StoreID = _T("");
57          m_UniqueID = _T("");
58          m_DiscountRate = 0.0;
59          m_DiscountAmount = 0.0;
60          //}}AFX_DATA_INIT
61  }
62
63
64  ////////////////////////////////////////////////////////////////////////////
65  // CCouponPropPage::DoDataExchange - Moves data between page and properties
66
67  void CCouponPropPage::DoDataExchange(CDataExchange* pDX)
68  {
69          //{{AFX_DATA_MAP(CCouponPropPage)
70          DDP_Text(pDX, IDC_EDIT1, m_StoreID, _T("StoreID") );
```

Oct 29 1996 16:06:20 CouponPpg.cpp Page 2

```
71          DDX_Text(pDX, IDC_EDIT1, m_StoreID);
72          DDP_Text(pDX, IDC_EDIT2, m_UniqueID, _T("UniqueID") );
73          DDX_Text(pDX, IDC_EDIT2, m_UniqueID);
74          DDP_Text(pDX, IDC_EDIT3, m_DiscountRate, _T("DiscountRate") );
75          DDX_Text(pDX, IDC_EDIT3, m_DiscountRate);
76          DDV_MinMaxDouble(pDX, m_DiscountRate, 0., 1.);
77          DDP_Text(pDX, IDC_EDIT4, m_DiscountAmount, _T("DiscountAmount") );
78          DDX_Text(pDX, IDC_EDIT4, m_DiscountAmount);
79          //}}AFX_DATA_MAP
80          DDP_PostProcessing(pDX);
81  }
82
83
84  ////////////////////////////////////////////////////////////////////////////
85  // CCouponPropPage message handlers
```

Oct 29 1996 16:08:20 CouponPpg.h Page 1

```
1   // CouponPpg.h : Declaration of the CCouponPropPage property page class.
2
3   ////////////////////////////////////////////////////////////////////////////
4   // CCouponPropPage : See CouponPpg.cpp.cpp for implementation.
5
6   class CCouponPropPage : public COlePropertyPage
7   {
8           DECLARE_DYNCREATE(CCouponPropPage)
9           DECLARE_OLECREATE_EX(CCouponPropPage)
10
11  // Constructor
12  public:
13          CCouponPropPage();
14
15  // Dialog Data
16          //{{AFX_DATA(CCouponPropPage)
17          enum { IDD = IDD_PROPPAGE_COUPON };
18          CString m_StoreID;
19          CString m_UniqueID;
20          double  m_DiscountRate;
21          double  m_DiscountAmount;
22          //}}AFX_DATA
23
24  // Implementation
25  protected:
26          virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
27
28  // Message maps
29  protected:
30          //{{AFX_MSG(CCouponPropPage)
31                  // NOTE - ClassWizard will add and remove member functions her
    e.
32                  //    DO NOT EDIT what you see in these blocks of generated co
    de !
33          //}}AFX_MSG
34          DECLARE_MESSAGE_MAP()
35
36  };
```

Oct 29 1996 16:06:21 StdAfx.cpp Page 1

```
1   // stdafx.cpp : source file that includes just the standard includes
2   //  stdafx.pch will be the pre-compiled header
3   //  stdafx.obj will contain the pre-compiled type information
4
5   #include "stdafx.h"
```

Oct 29 1996 16:06:21 **StdAfx.h** Page 1

```
1   // stdafx.h : include file for standard system include files,
2   //      or project specific include files that are used frequently,
3   //      but are changed infrequently
4
5   #define VC_EXTRALEAN            // Exclude rarely-used stuff from Windows head
    ers
6
7   #include <afxctl.h>         // MFC support for OLE Controls
8
9   // Delete the two includes below if you do not wish to use the MFC
10  //  database classes
11  #ifndef _UNICODE
12  #include <afxdb.h>                      // MFC database classes
13  #include <afxdao.h>                     // MFC DAO database classes
14  #endif //_UNICODE
```

Oct 29 1996 16:06:22 coupon.cpp Page 1

```
1   // coupon.cpp : Implementation of CCouponApp and DLL registration.
2
3   #include "stdafx.h"
4   #include "coupon.h"
5
6   #ifdef _DEBUG
7   #define new DEBUG_NEW
8   #undef THIS_FILE
9   static char THIS_FILE[] = __FILE__;
10  #endif
11
12
13  CCouponApp NEAR theApp;
14
15  const GUID CDECL BASED_CODE _tlid =
16                  { 0xebf68c60, 0x234a, 0x11d0, { 0xa0, 0x21, 0x44, 0x45, 0x53,
    0x54, 0, 0 } };
17  const WORD _wVerMajor = 1;
18  const WORD _wVerMinor = 0;
19
20
21  ////////////////////////////////////////////////////////////////////////////
22  // CCouponApp::InitInstance - DLL initialization
23
24  BOOL CCouponApp::InitInstance()
25  {
26          BOOL bInit = COleControlModule::InitInstance();
27
28          if (bInit)
29          {
30                  // TODO: Add your own module initialization code here.
31          }
32
33          return bInit;
34  }
35
36
37  ////////////////////////////////////////////////////////////////////////////
38  // CCouponApp::ExitInstance - DLL termination
39
40  int CCouponApp::ExitInstance()
41  {
42          // TODO: Add your own module termination code here.
43
44          return COleControlModule::ExitInstance();
45  }
46
47
48  ////////////////////////////////////////////////////////////////////////////
49  // DllRegisterServer - Adds entries to the system registry
50
51  STDAPI DllRegisterServer(void)
52  {
53          AFX_MANAGE_STATE(_afxModuleAddrThis);
54
55          if (!AfxOleRegisterTypeLib(AfxGetInstanceHandle(), _tlid))
56                  return ResultFromScode(SELFREG_E_TYPELIB);
57
58          if (!COleObjectFactoryEx::UpdateRegistryAll(TRUE))
59                  return ResultFromScode(SELFREG_E_CLASS);
60
61          return NOERROR;
62  }
63
64
65  ////////////////////////////////////////////////////////////////////////////
66  // DllUnregisterServer - Removes entries from the system registry
67
68  STDAPI DllUnregisterServer(void)
69  {
```

Oct 29 1996 16:06:22 coupon.cpp Page 2

```
70          AFX_MANAGE_STATE(_afxModuleAddrThis);
71
72          if (!AfxOleUnregisterTypeLib(_tlid))
73                  return ResultFromScode(SELFREG_E_TYPELIB);
74
75          if (!COleObjectFactoryEx::UpdateRegistryAll(FALSE))
76                  return ResultFromScode(SELFREG_E_CLASS);
77
78          return NOERROR;
79  }
```

Oct 29 1996 16:06:22 coupon.h Page 1

```
1   // coupon.h : main header file for COUPON.DLL
2
3   #if !defined( __AFXCTL_H__ )
4           #error include 'afxctl.h' before including this file
5   #endif
6
7   #include "resource.h"       // main symbols
8
9   /////////////////////////////////////////////////////////////////////////////
10  // CCouponApp : See coupon.cpp for implementation.
11
12  class CCouponApp : public COleControlModule
13  {
14  public:
15          BOOL InitInstance();
16          int ExitInstance();
17  };
18
19  extern const GUID CDECL _tlid;
20  extern const WORD _wVerMajor;
21  extern const WORD _wVerMinor;
```

Oct 29 1996 16:10:19 do.c Page 1

```
/*
 * do.c --
 *
 *      Digital Offer Core Library.
 *
 *  Copyright (c) 1995 Open Market, Inc.
 *  All rights reserved.
 *
 *  This file contains proprietary and confidential information and
 *  remains the unpublished property of Open Market, Inc. Use,
 *  disclosure, or reproduction is prohibited except as permitted by
 *  express written license agreement with Open Market, Inc.
 *
 *  $Id: do.c.v 1.7 1996/10/21 20:15:36 henry Exp $
 *
 *  Henry Luo
 *  henry@openmarket.com
 */

#ifndef lint
static char rcsid[] = "@(#) $Id: do.c.v 1.7 1996/10/21 20:15:36 henry Exp $";
#endif /* not lint */

#include "doint.h"
#include "global.h"
#include "md5.h"
#include "pdoint.h"


extern MSG_Control OSLDO_Messages;

/*
 *   Here are kind of internal routines
 */
```

Oct 29 1996 16:10:19 do.c Page 2

```
/*
  Return a string with "bad" characters escaped using the URL escaping
  mechanism (i.e. "%XX" where "XX" is the hex representation for the
  escaped character).
 */
static int OSL_urlEscape(char *inBuf, char *outBuf, int outBufLen, OSL_Error *
*e)
{
    char *inPtr = inBuf;
    int count = 0;
    unsigned int c;
    int rc;

    if (inBuf == NULL) {
        rc = OSLDO_E_NULL_INPUT_BUF;
        e->status = rc;
        sprintf(e->message, OS_Catgets(&OSLDO_Messages, rc));
        return (rc);
    }

    while ( c = (unsigned int) *(inPtr++) ) {
        if (c >= 128) {
            if (count >= outBufLen - 3 ) {
               *(outBuf+outBufLen-1) = 0;
               rc = OSLDO_E_BUF_OVERFLOW;
                    e->status = rc;
                    sprintf(e->message, OS_Catgets(&OSLDO_Messages, rc));
                    return (rc);
            }
            *(outBuf + count) = '%';
            sprintf( (outBuf+count+1), "%02x", c);
            count += 3;
            continue;
        }
        switch ( c ) {
            case '%':
            case '@':
            case '/':
            case '&':
            case '=':
            case ' ':
            case '+':
            case ':':
            case '?':
            case '>':
            case '<':
            case ';':
            case '{':
            case '\\':
            case '|':
            case '^':
            case '\'':
            case '"':
            case '[':
            case ']':
            case '}':
            case '~':
            case '`':
                if (count >= outBufLen - 3 ) {
                    *(outBuf+outBufLen-1) = 0;
                    rc = OSLDO_E_BUF_OVERFLOW;
                    e->status = rc;
                    sprintf(e->message, OS_Catgets(&OSLDO_Messages, rc));
                    return (rc);
                }
                *(outBuf + count) = '%';
                sprintf( (outBuf+count+1), "%02x", c);
                count += 3;
                break;
```

Oct 29 1996 16:10:19 do.c Page 3

```
105             default:
106                 if (count >= outBufLen - 1 ) {
107                     *(outBuf+outBufLen-1) = 0;
108                     rc = OSLDO_E_BUF_OVERFLOW;
109                     e->status = rc;
110                     sprintf(e->message, OS_Catgets(&OSLDO_Messages, rc));
111                     return (rc);
112                 }
113                 *(outBuf + count) = c;
114                 count += 1;
115                 break;
116         } /* end switch */
117     } /* end while */
118     *(outBuf + count) = 0; /* end of string */
119
120     return 0;
121 }
122
```

Oct 29 1996 16:10:19 do.c Page 4

```
123
124 /*
125  Given an array of name/value pairs, build a string in the following format:
126
127      name1=value1&name2=value2&name3=value3 ...
128
129   Note:  array element names should not contain "bad" charaters.
130  */
131 int WINAPI OSL_urlUnparse(OM_aa array, char *outBuf, int outBufLen, OSL_Error
*e)
132 {
133     char longBuf[OSL_MAX_BUF_LEN];
134     HashSearch searchPtr;
135     char el_name[OSL_MAX_HASH_KEY_NAME];
136     HashEntry *entry;
137     ClientData value;
138     char linkage = '&';
139     int count  = 0;
140     int n, rc;
141
142     for (entry = OM_aaFirstEntry(array, &searchPtr, el_name, &value);
143         entry != NULL;
144         entry = OM_aaNextEntry(array, &searchPtr, el_name, &value) ) {
145
146         if ( (rc = OSL_urlEscape((char *)value, longBuf, sizeof(longBuf),e)))
{
147             *(outBuf + count ) = 0;
148             return (rc);
149         }
150         n = strlen(longBuf) + strlen(el_name) + 2;
151         if ( (n + count) >= outBufLen) {
152             *(outBuf + count ) = 0;
153             rc  = (OSLDO_E_BUF_OVERFLOW);
154             e->status = rc ;
155             sprintf(e->message, OS_Catgets(&OSLDO_Messages, rc));
156             return (rc);
157         } else {
158             sprintf(outBuf+count, "%s=%s%c", el_name, longBuf, linkage);
159             count += n;
160         }
161     }
162     *(outBuf + count - 1) = 0;
163
164     return 0;
165
166 }
167
```

```
168
169 static int OSL_md5hash(char *src, char *outBuf,int outBufLen, OSL_Error *e)
170 {
171         MD5_CTX ctx;
172         char            *tp;
173         unsigned char   hash[16];
174         int             i;
175         int rc;
176
177         if (outBufLen < 33) {
178             rc = OSLDO_E_BUF_OVERFLOW;
179             e->status = rc;
180             sprintf(e->message, OS_Catgets(&OSLDO_Messages, rc));
181             return(rc);
182         }
183
184         OM_MD5Init(&ctx);
185         OM_MD5Update(&ctx,src,strlen(src));
186         OM_MD5Final(hash,&ctx);
187
188         tp = outBuf;
189         for(i=0;i<16;i++)
190                 {
191                 sprintf(tp,"%02x",hash[i]);
192                 tp += 2;
193                 }
194
195         return(0);
196 }
197
```



```
198
199 static int OSL_sign(char *content, char *key, char *ss, char *outBuf,
200         int outBufLen, OSL_Error *e)
201 {
202     char longBuf[OSL_MAX_BUF_LEN];
203     char hash[40];
204     int rc;
205
206     if ( strcmp("v1", ss) == 0 ) {
207         if ( ((strlen(content) + strlen(key) + 8) >= sizeof(longBuf)) ||
208              ((strlen(content) + 8 + 33 ) >= (size_t) outBufLen ) ) {
209             *outBuf = 0;
210             rc = OSLDO_E_BUF_OVERFLOW;
211             e->status = rc;
212             sprintf(e->message, OS_Catgets(&OSLDO_Messages, rc));
213             return (rc);
214         }
215
216         sprintf(longBuf, "%s ss=v1&%s", key, content);
217         OSL_md5hash(longBuf, hash, sizeof(hash), e);
218         sprintf(outBuf, "%s:ss=v1&%s", hash, content);
219         return (0);
220     } else if (strcmp("env", ss) == 0 ) {
221         if ( ((strlen(content) + strlen(key) + 9 + 64) >= sizeof(longBuf)) ||
222              ((strlen(content) + 9 +  33 ) >= (size_t) outBufLen ) ) {
223             *outBuf = 0;
224             rc = OSLDO_E_BUF_OVERFLOW;
225             e->status = rc;
226             sprintf(e->message, OS_Catgets(&OSLDO_Messages, rc));
227             return (rc);
228         }
229
230         sprintf(longBuf, "%-64s ss=env&%s %s", key, content, key);
231         OSL_md5hash(longBuf, hash, sizeof(hash), e);
232         sprintf(outBuf, "%s:ss=env&%s", hash, content);
233         return (0);
234     } else {
235         *outBuf = 0;
236         rc = OSLDO_E_UNKNOWN_SS;
237         e->status = rc;
238         sprintf(e->message, OS_Catgets(&OSLDO_Messages, rc), ss);
239         return (rc);
240     }
241
242 }
243
```

Oct 29 1996 16:10:19 do.c Page 7

```
244
245 int WINAPI OSL_mkpayload(char *kid, char *key, char *ss, OM_sa array, char *ou
    tBuf,
246         int outBufLen, OSL_Error *e)
247 {
248     int rc;
249     char *dummy;
250
251     char longBuf[OSL_MAX_BUF_LEN];
252
253     if (rc = OM_saAddEntry(array, "kid", (ClientData) strdup(kid)) ) {
254         sprintf(e->message, OS_Catgets(&OSLDO_Messages, rc));
255         return (rc);
256     }
257
258     if (dummy = (char *) OM_saGetEntry(array, "ss")) {
259         free (dummy);
260         OM_saDeleteEntry(array, "ss") ;
261     }
262
263     if ( rc = OSL_urlUnparse(array,  longBuf, sizeof(longBuf), e )) {
264         return (rc);
265     }
266
267     if (rc = OSL_sign(longBuf, key, ss, outBuf, outBufLen, e)) {
268         return (rc);
269     }
270
271     return (0);
272 }
273
274
```

Oct 29 1996 16:10:19 do.c Page 8

```
275
276 static int OSL_isAbsoluteUrl(char *url)
277 {
278
279     if ( strncmp(url, "http://", 7) == 0 ) return TRUE;
280     else if ( strncmp(url, "https://", 8) == 0 ) return TRUE;
281     else if ( strncmp(url, "shttp://", 8) == 0 ) return TRUE;
282     else if ( strncmp(url, "ftp://", 6) == 0 ) return TRUE;
283     else if ( strncmp(url, "mailto://", 9) == 0 ) return TRUE;
284     else if ( strncmp(url, "news://", 7) == 0 ) return TRUE;
285     else if ( strncmp(url, "gopher://", 9) == 0 ) return TRUE;
286     else if ( strncmp(url, "telnet://", 9) == 0 ) return TRUE;
287     else return FALSE;
288 }
289
290
291
```

```
292
293  OSL_Status WINAPI OSL_LoadKeyCacheFromKeyFile (OSL_KeyCache *keyCache,
294                              OSL_Error *error, OSL_Const_String keyfile,
295                              OSL_Const_String keytype)
296  {
297      int sts;
298      int rc = 0;
299
300      sts = omikdb_FFToKdb((char *)keyfile, (char *) keytype, (omikdb_kdb_p *) k
     eyCache);
301      if (sts != OMIKDB_SUCCESS)  {
302         rc = OSLDO_E_LOAD_KEYDB;
303         error->status = rc;
304         sprintf(error->message, OS_Catgets(&OSLDO_Messages, rc), keyfile);
305         return (rc);
306      }
307
308      return (rc);
309  }
310
311
312
```



```
313
314  OSL_Status WINAPI OSL_GetKeyFromKeyFile (OSL_Key *key,
315                  OSL_Error *error, OSL_Const_String keyType,
316                  OSL_Const_String storeID, int storeKeyID,
317                  time_t when, OSL_Const_String keyfile)
318  {
319      omikdb_kdb_p kdb;
320      int sts;
321      int rc = 0;
322
323      sts = omikdb_FFToKdb((char *)keyfile, (char *) keyType, &kdb);
324      if (sts != OMIKDB_SUCCESS)  {
325          rc = OSLDO_E_LOAD_KEYDB;
326          error->status = rc;
327          sprintf(error->message, OS_Catgets(&OSLDO_Messages, rc), keyfile);
328          return (rc);
329      }
330
331      rc = OSL_GetKeyFromKeyCache (key, error, keyType, storeID, storeKeyID,
332                  when, kdb);
333
334      omikdb_kdbFree(kdb);
335
336      return (rc);
337  }
338
339
```

```
340
341 OSL_Status WINAPI OSL_GetKeyFromKeyCache (OSL_Key *key,
342                 OSL_Error *error, OSL_Const_String keyType,
343                 OSL_Const_String storeID, int storeKeyID,
344                 time_t when, OSL_KeyCache keyCache)
345 {
346     omikdb_store_p store;
347     int sts;
348     int rc = 0;
349     char kid[50];
350     OSL_KeyStruct *keyObj;
351     omikdb_key_p keyPtr;
352
353     /*
354      *  when = 0 means current time
355      */
356     if (when == 0) when = time(0);
357
358     /*
359      *   The store has to be in the database
360      */
361     sts = omikdb_GetStore ((omikdb_kdb_p) keyCache, (char *) storeID, &store);
362     if (sts != OMIKDB_SUCCESS)  {
363         rc = OSLDO_E_GET_STORE_FM_KEYDB;
364         error->status = rc;
365         sprintf(error->message, OS_Catgets(&OSLDO_Messages, rc),
366                 storeID, ((omikdb_kdb_p) keyCache)->dbm_path);
367         return (rc);
368     }
369
370     /*
371      *   Offer key and receipt key is different
372      */
373     if ( strcmp(keyType, "0") == 0 ) {
374         /*  --- get offer key  --- */
375         if ( storeKeyID == 0) {
376             /*   Get offer key based on store id and time */
377             sts = omikdb_GetOfferKey(store, when, &keyPtr);
378         } else {
379             /*   Get offer key based on store id, store key id  and time */
380             sprintf(kid, "%s.%d", storeID, storeKeyID);
381             sts = omikdb_GetKidKey( (omikdb_kdb_p) keyCache, when, kid,&keyPtr
);
382         }
383     } else {
384         /*  --- get receipt key  --- */
385         sts = omikdb_GetValidateKey(store,when,storeKeyID, &keyPtr);
386     }
387
388     /*
389      *  handling errors
390      */
391     if (sts != OMIKDB_SUCCESS) {
392         switch (sts) {
393             case OMIKDB_KEYTOOEARLY:
394                 rc = OSLDO_E_KEY_TOO_EARLY;
395                 break;
396             case OMIKDB_KEYEXPIRED:
397                 rc = OSLDO_E_KEY_EXPIRED;
398                 break;
399             case OMIKDB_VALIDATEKEYNOTFOUND:
400             case OMIKDB_OFFERKEYNOTFOUND:
401             default:
402                 rc = OSLDO_E_KEY_NOT_FOUND;
403                 break;
404         }
405         error->status = rc;
406         sprintf(error->message, OS_Catgets(&OSLDO_Messages, rc), storeID,
407                 storeKeyID, ctime(&when));
```



```
408         return (rc);
409     }
410
411     /*
412      *  create and polulate key object
413      */
414     sprintf(kid, "%s.%d", storeID, keyPtr->skid);
415     keyObj  = (OSL_KeyStruct *) malloc (sizeof(OSL_KeyStruct));
416     if (key == NULL) {
417         rc = OSLDO_E_ALLOC_MEM;
418         error->status = rc;
419         sprintf(error->message, OS_Catgets(&OSLDO_Messages, rc));
420         return (rc);
421     }
422
423     keyObj->kid = strdup(kid);
424     keyObj->key = strdup(keyPtr->skey);
425     keyObj->signingScheme = strdup("env");
426     keyObj->begin = keyPtr->begin;
427     keyObj->end = keyPtr->end;
428     keyObj->expires = keyPtr->expires;
429
430     *key = (OSL_Key) keyObj;
431
432     /* everything should be OK */
433     return (0);
434 }
435
436
437 static int OSL_CopyServer(OSL_Server *dstServer, OSL_Error *error,
438         OSL_Server srcServer)
439 {
440     OSL_ServerStruct *server;
441
442     server = (OSL_ServerStruct *) srcServer;
443     return ( OSL_MakeServer (dstServer, error, server->scheme, server->host,
444                 server->port, server->script) );
445 }
446
447 static int OSL_CopyKey(OSL_Key *dstKey, OSL_Error *error, OSL_Key srcKey)
448 {
449     OSL_KeyStruct *keyObj;
450     OSL_KeyStruct *srcKeyObj;
451     int rc = 0;
452
453     srcKeyObj = (OSL_KeyStruct *)srcKey;
454
455     /*
456      *  create and polulate key object
457      */
458     keyObj  = (OSL_KeyStruct *) malloc (sizeof(OSL_KeyStruct));
459     if (keyObj == NULL) {
460         rc = OSLDO_E_ALLOC_MEM;
461         error->status = rc;
462         sprintf(error->message, OS_Catgets(&OSLDO_Messages, rc));
463         return (rc);
464     }
465
466     keyObj->kid = strdup(srcKeyObj->kid);
467     keyObj->key = strdup(srcKeyObj->key);
468     keyObj->signingScheme = strdup(srcKeyObj->signingScheme);
469     keyObj->begin = srcKeyObj->begin;
470     keyObj->end = srcKeyObj->end;
471     keyObj->expires = srcKeyObj->expires;
472
473     *dstKey = (OSL_Key) keyObj;
474
475     return (0);
476 }
477
```

Oct 29 1998 16:10:19 do.c Page 13

```
478 |
```

Oct 29 1998 16:10:19 do.c Page 14

```
479
480 int OSL_pdo2aa( OSL_Offer offer, OM_aa fields, OSL_Error *e)
481 {
482     int rc;
483
484     HashSearch searchPtr;
485     char el_name[OSL_MAX_HASH_KEY_NAME];
486     char fieldName[OSL_MAX_HASH_KEY_NAME];
487     HashEntry *entry;
488     OSL_OfferRow *slot;
489     OM_aa pdoSlot;
490     char tmpBuf[OSL_MAX_BUF_LEN];
491     char *value;
492     int cnt;
493     char *p;
494     char *oldVal;
495     OSL_OfferStruct *pdo;
496
497     pdo = (OSL_OfferStruct *) offer;
498
499     pdoSlot = &(pdo->rows);
500
501     /* --- set the value from slots in pdo to associative array  --- */
502     for (entry = OM_aaFirstEntry(pdoSlot, &searchPtr, el_name,
503         (ClientData *)&slot); entry != NULL;
504         entry = OM_aaNextEntry(pdoSlot, &searchPtr, el_name,
505         (ClientData *) &slot) ) {
506
507         if (slot->tag != NULL) {
508             strcpy(fieldName, slot->tag);
509            /*
510             *   IMPORTANT:
511             *   If the tag is "", it's not translated.
512             */
513             if (strlen(fieldName) == 0) continue;
514         } else {
515             strcpy(fieldName, el_name);
516         }
517
518         value = slot->value;
519         /*  --- if usedefault is set , try from default if value is not
520                 explict set --- */
521         if ( (slot->useDefault == 1)  && (value == NULL)  ) {
522             value = slot->_default;
523         }
524
525         if (value) {
526            /*
527             *  handle multi-level mapping
528             */
529             if ( p=strchr(fieldName, (int) '.') )  {
530                 *p = 0;
531                 cnt = 0;
532                 /*  --- if exists one with same parrent tag,
533                         get it --- */
534                 if ( oldVal = (char *) OM_aaGetEntry(fields, fieldName) ) {
535                     strcpy(tmpBuf, (char *) oldVal);
536                     cnt += strlen((char *) oldVal);
537                     tmpBuf[cnt] = '&';
538                     cnt ++;
539                 }
540                 /*  --- put the subname in --- */
541                 if ( rc = OSL_urlEscape(p+1, tmpBuf+cnt,
542                         (sizeof(tmpBuf) - cnt), e) ) {
543                         return (rc);
544                 }
545                 cnt = strlen(tmpBuf);
546                 tmpBuf[cnt] = '=';
547                 cnt ++;
548                 /*  --- put the value in --- */
```

```
549                 if ( rc = OSL_urlEscape(value, tmpBuf+cnt,
550                         (sizeof(tmpBuf) - cnt), e) ) {
551                         return (rc);
552                 }
553                 value = tmpBuf;
554             }
555
556             /*
557              *   Add an entry to the AA
558              */
559             if (rc = OM_aaAddEntry(fields, fieldName, (ClientData) strdup(valu
e)) ) {
560                 e->status = rc;
561                 sprintf(e->message, OS_Catgets(&OSLDO_Messages, rc));
562                 return (rc);
563             }
564         }
565     }
566
567     return (0);
568 }
569
570 static void OSL_SetIfEmpty(char **dst, char *src)
571 {
572     if ( *dst == NULL ) {
573         *dst = strdup(src);
574         return;
575     }
576
577     if (strlen(*dst) == 0 ) {
578         free (*dst);
579         *dst = strdup(src);
580         return;
581     }
582     return;
583 }
584
585 /* ----------------------------------------------------------------------
586 | ---------- ----------      --------- ---------------------------------
587 | --------- - --------- ----- -------- ---------------------------------
588 | -------- --- -------- ------ ------ ---------------------------------
589 | ------- ----- ------- ----- ------- ---------------------------------
590 | -------       -------      -------- ---------------------------------
591 | ------- ----- ------- ------------- ---------------------------------
592 | ------- ----- ------- ------------- ---------------------------------
593 | ------- ----- ------- ------------- ---------------------------------
594 |----------------------------------------------------------------------
595 |
596 |
597 /*
598  *    Here are the documented API
599  */
600
```



```
601 |
602 |/* ----------------------------------------------------------------------
603 |
604     NAME
605       OSL_WriteOfferToURL
606
607     DESCRIPTION
608       Given an offer described in the offer container OSL_Offer* offer,
609       create a Digital Offer in the string buffer URLbuf, at most 'buflen'
610       characters long.
611
612     PARAMETERS
613
614       OSL_Offer offer
615         An input argument, passed by reference, the offer to be created.
616         The names or tags in the offer will be used as the tags in the payload
617         The value or default value cells in the offer will be used as the valu
618         in the payload. Relative values, such as 'curl' will be made absolute
619         based on configuration values in the OSL_Store store.
620
621       OSL_Error* error
622         An output argument, * error points to the error object for
623         this call. The error object must already exist (allocated on
624         the heap or automatically in the caller's scope).
625
626       OSL_Store store
627         An input argument, passed by value, the store offering the article for
628         sale.
629
630       OSL_String URLbuf
631         An output argument, passed by reference, where the Digital Offer is
632         placed.
633
634       int buflen
635         An input argument, passed by value, the maximum lengthof the output
636         buffer URLbuf.
637
638     RETURN VALUES
639
640       OSL_NO_ERROR
641         Success.
642
643       OSLDO_E_CONTENT_HOST_NOT_SET
644         No known content host in the store. Occurs when OfferURL is set as a
645         relative URL, and hostname for content server is not set.
646
647       OSLDO_E_FULFILLMENT_HOST_NOT_SET
648         No known fulfillment host in the store.  Occurs when FulfillmentURL or
649         StatusURL is relative URL, and hostname for fulfillment server is not
650         set.
651
652       OSLDO_E_ALLOC_MEM
653         Can't allocate memory.
654
655       OSLDO_E_CREATE_HASH_ENTRY
656         Can't create hash entry. Host likly a memory problem.
657
658       [all the errors from OSL_SetOfferCell]
659       [all errors from OSL_CheckOffer]
660
661     SIDE EFFECTS
662       The URLbuf will be populated with the generated digital offer upon succe
663 |ss.
664 | ----------------------------------------------------------------------
665 | */
666 |OSL_Status WINAPI OSL_WriteOfferToURL (OSL_Offer offer, OSL_Error *error,
```

```
667                         OSL_Store store, OSL_String URLbuf, int buflen)
668 {
669
670     OM_aa fields;
671     int rc = 0;
672
673     OSL_StoreStruct * storeInfo;
674     char tmpBuf[OSL_MAX_BUF_LEN];
675     char *p;
676
677     int subscriptionFlag = 0;
678     int autoRenewFlag = 0;
679     int valid;
680     int result = 0;
681     int patchOffer = 0;
682     int len;
683
684     /*  cast to real thing */
685     storeInfo = (OSL_StoreStruct *) store;
686
687    /*
688     *   Check PDO constraints
689     */
690     if ( (rc = OSL_CheckOffer(offer, error)) != OSL_PASSED )  return (rc);
691
692     rc = 0;
693
694     if ( (fields = OM_aaCreate()) == NULL) {
695         rc = OSLDO_E_ALLOC_MEM;
696         error->status = rc;
697         sprintf(error->message, OS_Catgets(&OSLDO_Messages, rc));
698         return (rc);
699     }
700
701 #ifdef OSL_PATCH_OFFER
702
703
704    /*
705     *   IMPORTANT:
706     *
707     *   This is the current hack right now.  The trick used here will
708     *   prevent the ability of modifying the offer files (OSL.ofr) freely.
709     *   In the event of modifying OSL.ofr, special care needs to be taken
710     *   in accordence with the hacks below.
711     *   This hack does not apply to mdol (OMT.ofr).
712     */
713
714    /*
715     *   All happens if "Type" exists in the offer file.
716     */
717     if ( OSL_GetOfferCell(offer, error, "Type", OSL_Column_value,
718             tmpBuf, sizeof(tmpBuf)) == 0 ) {
719
720         patchOffer = 1;
721
722        /*
723         *    tangible => h
724         *    online   => i
725         */
726         if ( strcmp(tmpBuf, "tangible") == 0 )
727              strcpy(tmpBuf, "h");
728         else if (strcmp(tmpBuf, "online") == 0)
729              strcpy(tmpBuf, "i");
730
731         if ( rc = OSL_SetOfferCell(offer, error, "type", OSL_Column_value,
732              tmpBuf, 1) ) goto cleanup;
733
734        /*
735         *    prepend OfferURL with content server information if not
736         *    absolute url
```



```
737         */
738         if ( OSL_GetOfferCell(offer, error, "OfferURL", OSL_Column_value,
739              tmpBuf, sizeof(tmpBuf)) == 0 ) {
740             if ( !OSL_isAbsoluteUrl(tmpBuf) ) {
741
742                 /* --- error if content server is not configured --- */
743                 if (storeInfo->contentServer->host == NULL ||
744                     strlen(storeInfo->contentServer->host) == 0) {
745                     rc = OSLDO_E_CONTENT_HOST_NOT_SET;
746                     sprintf(error->message, OS_Catgets(&OSLDO_Messages, rc));
747                     error->status = rc;
748                     goto cleanup;
749                 }
750
751                 p = strdup(tmpBuf);
752                 /* prepend content server root */
753                 sprintf(tmpBuf, "%s://%s:%d", storeInfo->contentServer->scheme
754                         storeInfo->contentServer->host,
755                         storeInfo->contentServer->port);
756                 /* append path if avaiable */
757                 if (storeInfo->contentServer->script) {
758                     if ( *(storeInfo->contentServer->script) != '/' )
759                         strcat (tmpBuf, "/");
760                     strcat (tmpBuf, storeInfo->contentServer->script);
761                     len = strlen(tmpBuf);
762                     if ( tmpBuf[ len - 1 ] == '/' )
763                         tmpBuf[ len - 1 ] = '\0';
764                 }
765                 if ( *p != '/' ) strcat(tmpBuf, "/");
766                 strcat(tmpBuf, p);
767                 free(p);
768             }
769             if ( rc = OSL_SetOfferCell(offer, error, "curl",
770                 OSL_Column_value, tmpBuf, 1) )  goto cleanup;
771         }
772
773
774         /*
775          *   FulfillmentURL maps to different slot depending on
776          *   the value of Subscription
777          *
778          *   if Subscription == 1 then
779          *       FulfillmentURL => subs_url (detail.url)
780          *       url = http://subs.server/tms-subs-s/bin/subscription.cgi
781          *       fmt = get
782          *       AccountRequired = 1
783          *   else
784          *       FulfillmentURL => region_url (url)
785          *   endif
786          */
787         if ( OSL_GetOfferCell(offer, error, "Subscription",
788                 OSL_Column_valueDefault, tmpBuf, sizeof(tmpBuf)) == 0 ) {
789             if (strcmp(tmpBuf, "1") == 0) subscriptionFlag = 1;
790             else subscriptionFlag = 0;
791         }
792
793         /*
794          *    prepend OfferURL with fulfillment server information if not
795          *    absolute url
796          */
797         if ( OSL_GetOfferCell(offer, error, "FulfillmentURL", OSL_Column_value
798              tmpBuf, sizeof(tmpBuf)) == 0 ) {
799             if ( !OSL_isAbsoluteUrl(tmpBuf) ) {
800
801                 /* --- error if fulfillment server is not configured --- */
802                 if (storeInfo->fulfillmentServer->host == NULL ||
803                     strlen(storeInfo->fulfillmentServer->host) == 0) {
804                     rc = OSLDO_E_FULFILLMENT_HOST_NOT_SET;
```

Oct 29 1996 16:10:19 **do.c** Page 19

```
805                     sprintf(error->message, OS_Catgets(&OSLDO_Messages, rc));
806                     error->status = rc;
807                     goto cleanup;
808                 }
809
810                 p = strdup(tmpBuf);
811                 /* prepend fulfillment server root */
812                 sprintf(tmpBuf, "%s://%s:%d",
813                         storeInfo->fulfillmentServer->scheme,
814                         storeInfo->fulfillmentServer->host,
815                         storeInfo->fulfillmentServer->port);
816                 /* append path if avaiable */
817                 if (storeInfo->fulfillmentServer->script) {
818                     if ( *(storeInfo->fulfillmentServer->script) != '/' )
819                        strcat (tmpBuf, "/");
820                     strcat (tmpBuf, storeInfo->fulfillmentServer->script);
821                     len = strlen(tmpBuf);
822                     if ( tmpBuf[ len - 1 ] == '/' )
823                        tmpBuf[ len - 1 ] = '\0';
824                 }
825                 if ( *p != '/' ) strcat(tmpBuf, "/");
826                 strcat(tmpBuf, p);
827                 free(p);
828
829             }  /* end if isAbsoluteURL */
830
831             if (subscriptionFlag == 0) {
832                 if ( rc = OSL_SetOfferCell(offer, error, "region_url",
833                         OSL_Column_value, tmpBuf, 1) )  goto cleanup;
834             } else {
835                 if ( rc = OSL_SetOfferCell(offer, error, "subs_url",
836                         OSL_Column_value, tmpBuf, 1) )  goto cleanup;
837                 sprintf(tmpBuf, "%s://%s:%d%s",
838                         storeInfo->subscriptionServer->scheme,
839                         storeInfo->subscriptionServer->host,
840                         storeInfo->subscriptionServer->port,
841                         storeInfo->subscriptionServer->script);
842
843                 if ( rc = OSL_SetOfferCell(offer, error, "url",
844                         OSL_Column_value, tmpBuf, 1) ) goto cleanup;
845                 if ( rc = OSL_SetOfferCell(offer, error, "fmt",
846                         OSL_Column_value, "get", 1) ) goto cleanup;
847             }  /* end if subscriptionFlag == 0 */
848         }  /* end if get FulfillmentURL value */
849
850
851         /*
852          *   prepend StatusURL with fulfillment server information if not
853          *   absolute url
854          */
855         if ( OSL_GetOfferCell(offer, error, "StatusURL", OSL_Column_value,
856                 tmpBuf, sizeof(tmpBuf)) == 0 ) {
857             if ( !OSL_isAbsoluteUrl(tmpBuf) ) {
858
859                 /* --- error if fulfillment server is not configured --- */
860                 if (storeInfo->fulfillmentServer->host == NULL ||
861                     strlen(storeInfo->fulfillmentServer->host) == 0) {
862                     rc = OSLDO_E_FULFILLMENT_HOST_NOT_SET;
863                     sprintf(error->message, OS_Catgets(&OSLDO_Messages, rc));
864                     error->status = rc;
865                     goto cleanup;
866                 }
867
868                 p = strdup(tmpBuf);
869                 /* prepend fulfillment server root */
870                 sprintf(tmpBuf, "%s://%s:%d",
871                         storeInfo->fulfillmentServer->scheme,
872                         storeInfo->fulfillmentServer->host,
873                         storeInfo->fulfillmentServer->port);
874                 /* append path if avaiable */
```

Oct 29 1996 16:10:19 **do.c** Page 20

```
875                 if (storeInfo->fulfillmentServer->script) {
876                     if ( *(storeInfo->fulfillmentServer->script) != '/' )
877                        strcat (tmpBuf, "/");
878                     strcat (tmpBuf, storeInfo->fulfillmentServer->script);
879                     len = strlen(tmpBuf);
880                     if ( tmpBuf[ len - 1 ] == '/' )
881                        tmpBuf[ len - 1 ] = '\0';
882                 }
883                 if ( *p != '/' ) strcat(tmpBuf, "/");
884                 strcat(tmpBuf, p);
885                 free(p);
886             }
887             if ( rc = OSL_SetOfferCell(offer, error, "status_url",
888                     OSL_Column_value, tmpBuf, 1) )  goto cleanup;
889         }
890
891         /*
892          *   SubscriptionDuration maps to different tags depending on
893          *   the value of AutoRenew
894          *
895          *   if AutoRenew == 1 then
896          *       SubscriptionDuration => renew
897          *   else
898          *       SubscriptionDuration => subs_duration (detail.duration)
899          *   endif
900          */
901         if ( subscriptionFlag ) {
902             strcpy(tmpBuf, "");
903             if ( OSL_GetOfferCell(offer, error, "AutoRenew",
904                     OSL_Column_valueDefault, tmpBuf, sizeof(tmpBuf)) == 0)
905                 if (strcmp(tmpBuf, "1") == 0) autoRenewFlag = 1;
906
907             if (rc = OSL_GetOfferCell(offer, error, "SubscriptionDuration",
908                     OSL_Column_valueDefault, tmpBuf, sizeof(tmpBuf)) )
909                 return (rc);
910             if (autoRenewFlag) {
911                 if ( rc = OSL_SetOfferCell(offer, error, "renew",
912                         OSL_Column_value, tmpBuf, 1) ) goto cleanup;
913             } else {
914                 if ( rc = OSL_SetOfferCell(offer, error, "subs_duration",
915                         OSL_Column_value, tmpBuf, 1) ) goto cleanup;
916             }
917         }
918
919     }  /* end of "Type" */
920
921 #endif  /* OSL_PATCH_OFFER  */
922
923     /*
924      *  Check -valid field against current time
925      */
926     if ( OSL_GetOfferCell(offer, error, "OfferExpires", OSL_Column_value,
927             tmpBuf, sizeof(tmpBuf)) == 0 ) {
928         valid = atoi(tmpBuf);
929         if ( valid <  time(NULL) ) {
930             rc = OSLDO_E_EXPIRES_BEFORE_NOW;
931             error->status = rc;
932             sprintf(error->message, OS_Catgets(&OSLDO_Messages, rc));
933             goto cleanup;
934         }
935     }
936
937     /*
938      *   Pdo to AA
939      */
940
941     if ( rc = OSL_pdo2aa(offer, fields, error) ) {
942         goto cleanup;
943     }
944
```

Oct 29 1996 16:10:19 do.c Page 21

```
945 #ifdef OSL_PATCH_OFFER
946     if (patchOffer && subscriptionFlag) {
947         if (rc = OM_aaAddEntry(fields, "acctreqd", (ClientData) strdup("1")) )
    {
948             error->status = rc;
949             sprintf(error->message, OS_Catgets(&OSLDO_Messages, rc));
950             goto cleanup;
951         }
952     }
953 #endif /* OSL_PATCH_OFFER */
954
955     /*
956      *   Generate the payload
957      */
958
959     if ( rc = OSL_mkpayload(storeInfo->key->kid, storeInfo->key->key,
960         storeInfo->key->signingScheme, fields, tmpBuf, sizeof(tmpBuf), error)
    ) {
961         goto cleanup;
962     }
963
964     /*
965      *   construct DO  check length of out buf
966      */
967     if ( (strlen(storeInfo->transactServer->scheme) +
968           strlen(storeInfo->transactServer->host) +
969           strlen(storeInfo->transactServer->script) +
970           strlen(tmpBuf) + 5 + 8 )  >= (size_t)buflen )  {
971         rc = OSLDO_E_BUF_OVERFLOW;
972         error->status = rc;
973         sprintf(error->message, OS_Catgets(&OSLDO_Messages, rc));
974         goto cleanup;
975     }
976
977     sprintf(URLbuf, "%s://%s:%d%s?%s", storeInfo->transactServer->scheme,
978         storeInfo->transactServer->host, storeInfo->transactServer->port,
979         storeInfo->transactServer->script, tmpBuf);
980
981     /*
982      *   clean up
983      */
984 cleanup:
985
986     OM_aaDelete(fields);
987
988 #ifdef OSL_PATCH_OFFER
989     result = rc;
990
991     if (patchOffer) {
992     if ( rc = OSL_SetOfferCell(offer, error, "subs_duration",
993                         OSL_Column_value, NULL, 1) ) return (rc);
994     if ( rc = OSL_SetOfferCell(offer, error, "type",
995                         OSL_Column_value, NULL, 1) ) return (rc);
996     if ( rc = OSL_SetOfferCell(offer, error, "curl",
997                         OSL_Column_value, NULL, 1) ) return (rc);
998     if ( rc = OSL_SetOfferCell(offer, error, "region_url",
999                         OSL_Column_value, NULL, 1) ) return (rc);
1000    if ( rc = OSL_SetOfferCell(offer, error, "subs_url",
1001                        OSL_Column_value, NULL, 1) ) return (rc);
1002    if ( rc = OSL_SetOfferCell(offer, error, "url",
1003                        OSL_Column_value, NULL, 1) ) return (rc);
1004    if ( rc = OSL_SetOfferCell(offer, error, "fmt",
1005                        OSL_Column_value, NULL, 1) ) return (rc);
1006    if ( rc = OSL_SetOfferCell(offer, error, "renew",
1007                        OSL_Column_value, NULL, 1) ) return (rc);
1008    if ( rc = OSL_SetOfferCell(offer, error, "status_url",
1009                        OSL_Column_value, NULL, 1) ) return (rc);
1010    if ( rc = OSL_SetOfferCell(offer, error, "subs_duration",
1011                        OSL_Column_value, NULL, 1) ) return (rc);
1012    }
```

Oct 29 1996 16:10:19 do.c Page 22

```
1013 #endif /* OSL_PATCH_OFFER */
1014
1015     return (result);
1016 }
1017
```

```
1018
1019 /* ============================================================================
     *=
1020
1021     NAME
1022       OSL_LoadKeyCacheFromFile
1023
1024     DESCRIPTION
1025       Load keys found in a flat key file into the memory cache for later use.
1026
1027     PARAMETERS
1028
1029       OSL_KeyCache* keycache
1030         Output argument, passed by reference. Creates an OSL_KeyCache object
1031         by reading in the contents of the keyfile.
1032
1033       OSL_Error* error
1034         An output argument, * error points to the error object for
1035         this call. The error object must already exist (allocated on
1036         the heap or automatically in the caller's scope). Populated only
1037         on error.
1038
1039       OSL_Const_String keyfile
1040         An input argument, the filename of the keyfile. The file
1041         must exist and be readable to the process issuing this call.
1042         The filename can be absolute or relative.
1043
1044     RETURN VALUES
1045
1046       OSL_NO_ERROR
1047         Success.
1048
1049       OSLDO_E_LOAD_KEYDB
1050         Could not read the keyfile into a cache object.
1051
1052     SIDE EFFECTS
1053       A OSL_KeyCache Object will be created on the heap.
1054  ----------------------------------------------------------------------------
1055  */
1056 OSL_Status WINAPI OSL_LoadKeyCacheFromFile (OSL_KeyCache *keyCache,
1057                                             OSL_Error *error, OSL_Const_String keyfile)
1058 {
1059     return (OSL_LoadKeyCacheFromKeyFile (keyCache, error, keyfile, '0'));
1060 }
1061
```



```
1062
1063 /* ============================================================================
     *=
1064
1065     NAME
1066       OSL_FreeKeyCache
1067
1068     DESCRIPTION
1069       Delete a key cache when it is no longer needed.
1070
1071     PARAMETERS
1072
1073       OSL_KeyCache* keyCache
1074         An input argument, passed by reference, the key cache to  delete.
1075
1076     RETURN VALUES
1077       None.
1078  ----------------------------------------------------------------------------
1079  */
1080 void WINAPI OSL_FreeKeyCache (OSL_KeyCache *keyCache)
1081 {
1082     if (*keyCache == NULL) return;
1083     omikdb_kdbFree((omikdb_kdb_p) *keyCache);
1084     *keyCache = NULL;
1085     return;
1086 }
1087
```


WO 98/19224 PCT/US97/19391

```
1088
1089 /* ************************************************************************
*-
1090
1091     NAME
1092       OSL_GetKeyFromFile
1093
1094     DESCRIPTION
1095       Get a valid key from a flat key file for a particular store.
1096
1097     PARAMETERS
1098
1099       OSL_Key* key
1100         Output argument, passed by reference, allocated if a key can
1101         be found for the store id 'storeID' in the OSL_KeyCache. The key
1102         must be freed with OSL_FreeKey.
1103
1104       OSL_Error* error
1105         An output argument, * error points to the error object for
1106         this call. The error object must already exist (allocated on
1107         the heap or automatically in the caller's scope). Populated only
1108         on error.
1109
1110       OSL_Const_String storeID
1111         An input argument, passed by read only reference. The store id
1112         who's key you want to find.
1113
1114       OSL_Const_String keyFile
1115         An input argument, passed by read only reference, the name of
1116         the key file to search.
1117
1118     RETURN VALUES
1119
1120       OSL_NO_ERROR
1121         Success.
1122
1123       OSLDO_E_LOAD_KEYDB
1124         Keyfile is not found or corrupted.
1125
1126       OSLDO_E_GET_STORE_FM_KEYDB
1127         This store is not in the key file.
1128
1129       OSLDO_E_KEY_TOO_EARLY
1130         No key is available for MACing yet.
1131
1132       OSLDO_E_KEY_EXPIRED
1133         No key is active anymore.
1134
1135       OSLDO_E_KEY_NOT_FOUND
1136         No key for this store whatsoever.
1137
1138       OSLDO_E_ALLOC_MEM
1139         Could not allocate memory for the OSL_Key object.
1140
1141     SIDE EFFECTS
1142       A OSL_Key object will be created and populated on the heap upon success.
1143
1144 ------------------------------------------------------------------------
1145  */
1146 OSL_Status WINAPI OSL_GetKeyFromFile (OSL_Key *key,
1147                 OSL_Error *error, OSL_Const_String storeID,
1148                 OSL_Const_String keyfile)
1149 {
1150     return(OSL_GetKeyFromKeyFile (key, error, "O", storeID, 0, 0, keyfile));
1151 }
1152
1153
```



```
1154
1155 /* ************************************************************************
*-
1156
1157     NAME
1158       OSL_GetKeyFromCache
1159
1160     DESCRIPTION
1161       Get a valid key from key cache for a particular store.
1162
1163     PARAMETERS
1164
1165       OSL_Key* key
1166         Output argument, passed by reference, allocated if a key can
1167         be found for the store id 'storeID' in the OSL_KeyCache. The key
1168         must be freed with OSL_FreeKey.
1169
1170       OSL_Error* error
1171         An output argument, * error points to the error object for
1172         this call. The error object must already exist (allocated on
1173         the heap or automatically in the caller's scope). Populated only
1174         on error.
1175
1176       OSL_Const_String storeID
1177         An input argument, passed by read only reference. The store id
1178         who's key you want to find.
1179
1180       OSL_KeyCache keyCache
1181         An input argument, passed by value, the key cache holding the
1182         keys.
1183
1184     RETURN VALUES
1185
1186       OSL_NO_ERROR
1187         Success.
1188
1189       OSLDO_E_GET_STORE_FM_KEYDB
1190         This store is not in the key file.
1191
1192       OSLDO_E_KEY_TOO_EARLY
1193         No key is available for MACing yet.
1194
1195       OSLDO_E_KEY_EXPIRED
1196         No key is active anymore.
1197
1198       OSLDO_E_KEY_NOT_FOUND
1199         No key for this store whatsoever.
1200
1201       OSLDO_E_ALLOC_MEM
1202         Could not allocate memory for the OSL_Key object.
1203
1204     SIDE EFFECTS
1205       A OSL_Key object will be created and populated on the heap upon success.
1206
1207 ------------------------------------------------------------------------
1208  */
1209 OSL_Status WINAPI OSL_GetKeyFromCache (OSL_Key *key, OSL_Error *error,
1210                 OSL_Const_String storeID, OSL_KeyCache keyCache)
1211 {
1212     return ( OSL_GetKeyFromKeyCache (key, error, "O", storeID,
1213                 0, 0, keyCache) );
1214 }
1215
```


WO 98/19224 -49- PCT/US97/19391

```
1216
1217 /* ------------------------------------------------------------------------
     +*
1218
1219     NAME
1220       OSL_FreeKey
1221
1222     DESCRIPTION
1223       Delete a key when it is no longer needed.
1224
1225     PARAMETERS
1226
1227       OSL_Key* key
1228         An input argument, passed by reference, the key to delete.
1229
1230     RETURN VALUES
1231       None.
1232
1233  ------------------------------------------------------------------------
1234  */
1235 void WINAPI OSL_FreeKey(OSL_Key *key)
1236 {
1237     OSL_KeyStruct *keyObj;
1238
1239     if ( *key == NULL) return;
1240
1241     keyObj = (OSL_KeyStruct *) *key;
1242     if (keyObj->kid) free(keyObj->kid);
1243     if (keyObj->key) free(keyObj->key);
1244     if (keyObj->signingScheme) free(keyObj->signingScheme);
1245     free(keyObj);
1246
1247     *key = NULL;
1248
1249     return;
1250 }
1251
1252
```



```
1253
1254 /* ------------------------------------------------------------------------
     +*
1255
1256     NAME
1257       OSL_MakeServer
1258
1259     DESCRIPTION
1260       OSL_MakeServer makes an OSL_Server object out of the *constituent pieces
     +*
1261       of a URL referencing the server. For example, if a server had the URL
1262       *http://my.host.com:8080/foo/bar/smoo.cgi*, then its constituent
1263       pieces would be *http*, *my.host.com*, 8080 and */foo/bar/smoo.cgi*.
1264       These are parts of an OSL_Server object and are passed as input to
1265       OSL_MakeServer.
1266
1267     PARAMETERS
1268
1269       OSL_Server* server
1270         An output argument, passed by reference, the server object allocated a
     +nd
1271         populated in heap storage. This object must be freed using
1272         OSL_FreeServer.  (Note that OSL_Server is actually a reference type,
1273         so the allocated object is, indeed, returned correctly.)
1274
1275       OSL_Error* error
1276         An output argument, error points to the error object for
1277         this call. The error object must already exist (allocated on
1278         the heap or automatically in the caller's scope). Error is
1279         populated only in the event of an error.
1280
1281       OSL_Const_String scheme
1282         An input argument, passed by read only reference. The scheme can be
1283         either *http* or *https*, lowercase only.
1284
1285       OSL_Const_String host
1286         An input argument, passed by read only reference. The host can be
1287         either an internet host name or IP addressed in *dotted quad* notation
     +.
1288         Host names are case insensitive.
1289
1290       int port
1291         An input argument, passed by value, the port for the IP host above. Mu
     +st
1292         be nonnegative. If port is 0, then port is set to the default port
1293         depending on the scheme: 80 for http or 443 for https.
1294
1295       OSL_Const_String script
1296         An input argument, passed by read only reference, the script that will
1297         deliver the service requested. Case sensitivity often depends on the
1298         operating system that the host is running, but usually case is sensiti
     +ve.
1299         A NULL value means no script is named.
1300
1301     RETURN VALUES
1302
1303       OSL_NO_ERROR
1304         Success.
1305
1306       OSLDO_E_ALLOC_MEM
1307         The server object was not created because the heap is exhausted.
1308
1309       OSLDO_E_WRONG_SCHEME
1310         The scheme was something other than *http* or *https*.
1311
1312     SIDE EFFECTS
1313       A OSL_Server object will be created on the heap upon success.
1314
1315  ------------------------------------------------------------------------
1316  */
```

```
1317 OSL_Status WINAPI OSL_MakeServer(OSL_Server *server, OSL_Error *error,
1318         OSL_Const_String scheme, OSL_Const_String host, int port,
1319         OSL_Const_String script)
1320 {
1321     OSL_ServerStruct *serverObj;
1322     int rc = 0;
1323
1324     /*
1325      *  port number cannot be negative, zero is OK for default
1326      */
1327     if (port < 0) {
1328         rc = OSLDO_E_INVALID_PORT;
1329         error->status = rc;
1330         sprintf(error->message, OS_Catgets(&OSLDO_Messages, rc), port);
1331         return (rc);
1332     }
1333
1334     /*
1335      *  allocate memory
1336      */
1337     serverObj  = (OSL_ServerStruct *) malloc (sizeof(OSL_ServerStruct));
1338     if (serverObj == NULL) {
1339         rc = OSLDO_E_ALLOC_MEM;
1340         error->status = rc;
1341         sprintf(error->message, OS_Catgets(&OSLDO_Messages, rc));
1342         return (rc);
1343     }
1344
1345     /*
1346      *  assign scheme
1347      */
1348     if (scheme == NULL || strlen(scheme) == 0) {
1349         serverObj->scheme = strdup("http");
1350     } else {
1351         if ( strcmp(scheme, "http") != 0 && strcmp(scheme, "https") != 0 ) {
1352             rc = OSLDO_E_WRONG_SCHEME;
1353             sprintf(error->message, OS_Catgets(&OSLDO_Messages, rc), scheme);
1354             error->status = rc;
1355             free (serverObj);
1356             return (rc);
1357         } else {
1358             serverObj->scheme = strdup(scheme);
1359         }
1360     }
1361
1362     /*
1363      *  assign host
1364      */
1365     if (host != NULL)  serverObj->host = strdup(host);
1366     else serverObj->host = NULL;
1367
1368     /*
1369      *  assign port
1370      */
1371     if (port != 0)  serverObj->port = port;
1372     else if (strcmp(serverObj->scheme, "https") == 0) serverObj->port = 443;
1373     else serverObj->port = 80;
1374
1375     /*
1376      *  assign cgi script
1377      */
1378     if (script != NULL) serverObj->script = strdup(script);
1379     else serverObj->script = NULL;
1380
1381     *server = (OSL_Server ) serverObj;
1382
1383     return (0);
1384 }
1385
```



```
1386
1387 /* ************************************************************************
     **
1388
1389     NAME
1390       OSL_FreeServer
1391
1392     DESCRIPTION
1393       Free a server's memory when it is no longer needed.
1394
1395     PARAMETERS
1396
1397       OSL_Server* server
1398         An input argument, passed by reference, the server to  free.
1399
1400     RETURN VALUES
1401       None.
1402
1403  ------------------------------------------------------------------------
1404  */
1405 void WINAPI OSL_FreeServer(OSL_Server *server)
1406 {
1407     OSL_ServerStruct *serverObj;
1408
1409     if ( *server == NULL ) return;
1410     serverObj = (OSL_ServerStruct *) *server;
1411
1412     if ( serverObj->scheme ) free (serverObj->scheme);
1413     if ( serverObj->host ) free (serverObj->host);
1414     if ( serverObj->script ) free (serverObj->script);
1415     free (serverObj);
1416     *server = NULL;
1417     return;
1418 }
1419
```

```
1420
1421 /* ============================================================================
+*
1422
1423     NAME
1424       OSL_MakeStore
1425
1426     DESCRIPTION
1427       OSL_MakeStore allocates  a new store object on the heap. This object
1428       is populated with copies of each of the component objects, namely the
1429       server objects and the key. The store must be freed using OSL_FreeStore.
1430       Each component must also be freed using OSL_FreeServer and OSL_FreeKey.
1431       A store is usually allocated to be passed as input to OSL_WriteOfferToUR
+L.
1432
1433     PARAMETERS
1434
1435       OSL_Server* store
1436         An output argument, passed by reference. If the call succeeds, store
1437         points to newly allocated memory in heap storage containing a store.
1438         'Store' must be freed by OSL_FreeStore.
1439
1440       OSL_Error* error
1441         An output argument, * error points to the error object for
1442         this call. The error object must already exist (allocated on
1443         the heap or automatically in the caller's scope). Only populated
1444         on error.
1445
1446       OSL_Const_String storeID
1447         An input argument, passed by read only reference, containing the
1448         store id for this store, usually a large, positive integer.
1449
1450       OSL_Server transactServer
1451         An input argument, passed by reference.  Must not be NULL.
1452
1453       OSL_Server fulfillmentServer
1454         An input argument, passed by reference, the fulfillment server  Must
1455         not be NNULL.
1456
1457       OSL_Server contentServer
1458         An input argument, passed by reference, the content server  Allows no
1459         default argument.  Must not be NULL.
1460
1461       OSL_Server subscriptionServer
1462         An input argument, passed by reference. Must not be NULL.
1463
1464       OSL_Key key
1465         An input argument, passed by reference, the key this store will use fo
+r
1466         Digital Offers. Accepts no defaults.
1467
1468     RETURN VALUES
1469
1470       OSL_NO_ERROR
1471         Success.
1472
1473       OSLDO_E_ALLOC_MEM
1474         The store object was not created because the heap is exhausted.
1475
1476     SIDE EFFECTS
1477       A OSL_Store object will be created on the heap upon success.
1478
1479 ----------------------------------------------------------------------------
1480  */
1481 OSL_Status WINAPI OSL_MakeStore(OSL_Store *store, OSL_Error *error,
1482         OSL_Const_String storeID, OSL_Server transactServer,
1483         OSL_Server fulfillmentServer, OSL_Server contentServer,
1484         OSL_Server subscriptionServer, OSL_Key key)
1485 {
1486     OSL_StoreStruct *storeObj = NULL;
```



```
1487     int rc = 0;
1488     int subsDefaulted = FALSE;
1489     OSL_KeyStruct *keyStruct;
1490
1491     keyStruct = (OSL_KeyStruct *) key;
1492
1493     /*
1494      *  Check if storeID matches the first half of kid in Key Object
1495      */
1496     if ( strncmp(storeID, keyStruct->kid, strlen(storeID)) != 0 ) {
1497         rc = OSLDO_E_STOREID_KID_NOMATCH;
1498         error->status = rc;
1499         sprintf(error->message, OS_Catgets(&OSLDO_Messages, rc), storeID, keyS
+truct->kid);
1500         return (rc);
1501     }
1502
1503     /*
1504      *  allocate memory
1505      */
1506     storeObj  = (OSL_StoreStruct *) malloc (sizeof(OSL_StoreStruct));
1507     if (storeObj == NULL) {
1508         rc = OSLDO_E_ALLOC_MEM;
1509         error->status = rc;
1510         sprintf(error->message, OS_Catgets(&OSLDO_Messages, rc));
1511         return (rc);
1512     }
1513
1514     /*
1515      *  Initialize the StoreStruct
1516      */
1517     storeObj->transactServer = NULL;
1518     storeObj->fulfillmentServer = NULL;
1519     storeObj->contentServer = NULL;
1520     storeObj->subscriptionServer = NULL;
1521     storeObj->key = NULL;
1522     storeObj->storeID = NULL;
1523
1524
1525     /*
1526      *  copy the transact server
1527      */
1528     if (rc=OSL_CopyServer((OSL_Server *) &(storeObj->transactServer),error,
1529         transactServer)) goto ErrorMakeStore;
1530     OSL_SetIfEmpty(&(storeObj->transactServer->host), "payment.openmarket.com"
+);
1531     OSL_SetIfEmpty(&(storeObj->transactServer->script), "/tms-ts/bin/payment.c
+gi");
1532
1533     /*
1534      *  copy the subscription server
1535      */
1536     /*  Defaults to transact server is NULL is passed in */
1537     if (subscriptionServer == NULL) {
1538         subscriptionServer = transactServer;
1539         subsDefaulted = TRUE;
1540     }
1541     if (rc=OSL_CopyServer((OSL_Server *) &(storeObj->subscriptionServer),error
+,
1542         subscriptionServer)) goto ErrorMakeStore;
1543     OSL_SetIfEmpty(&(storeObj->subscriptionServer->host),"payment.openmarket.c
+om");
1544     OSL_SetIfEmpty(&(storeObj->subscriptionServer->script), "/tms-subs-s/bin/s
+ubscription.cgi");
1545     if (subsDefaulted) {
1546         free ( storeObj->subscriptionServer->script );
1547         storeObj->subscriptionServer->script =
1548                 strdup("/tms-subs-s/bin/subscription.cgi");
1549     }
1550
```

```
1551
1552        /*
1553         *  copy the content server
1554         */
1555         if (rc=OSL_CopyServer((OSL_Server *) &(storeObj->contentServer),error,
1556             contentServer)) goto ErrorMakeStore;
1557
1558        /*
1559         *  copy the Fulfillment server
1560         */
1561         /* defaults to content server */
1562         if (fulfillmentServer == NULL) fulfillmentServer = contentServer;
1563
1564         if (rc=OSL_CopyServer((OSL_Server *) &(storeObj->fulfillmentServer),error,
1565             fulfillmentServer)) goto ErrorMakeStore;
1566
1567        /*
1568         *  copy the key
1569         */
1570         if (rc = OSL_CopyKey( (OSL_Key *) &(storeObj->key), error, key ) )
1571             goto ErrorMakeStore;
1572
1573        /*
1574         *  give the store ID
1575         */
1576         storeObj->storeID = strdup(storeID);
1577
1578        /*
1579         *  set the output
1580         */
1581         *store = (OSL_Store) storeObj;
1582         return (0);
1583
1584    ErrorMakeStore:
1585         OSL_FreeStore ( (OSL_Store *) &storeObj);
1586         return (rc);
1587    }
1588
```



```
1589
1590    /* ========================================================================
     +=
1591
1592         NAME
1593           OSL_FreeStore
1594
1595         DESCRIPTION
1596           Delete a store when it is no longer needed.
1597
1598         PARAMETERS
1599
1600           OSL_Store* store
1601             An input argument, passed by reference, the store to  delete.
1602
1603         RETURN VALUES
1604           None.
1605
1606     ------------------------------------------------------------------------
1607     */
1608    void WINAPI OSL_FreeStore(OSL_Store *store)
1609    {
1610         OSL_StoreStruct *storeObj;
1611
1612         if ( *store == NULL ) return;
1613         storeObj = (OSL_StoreStruct *) *store;
1614
1615         OSL_FreeServer( (OSL_Server *) &( storeObj->transactServer ) );
1616         OSL_FreeServer( (OSL_Server *) &( storeObj->fulfillmentServer ) );
1617         OSL_FreeServer( (OSL_Server *) &( storeObj->contentServer ) );
1618         OSL_FreeServer( (OSL_Server *) &( storeObj->subscriptionServer ) );
1619         OSL_FreeKey( (OSL_Key *) &( storeObj->key ) );
1620         if (storeObj->storeID) free (storeObj->storeID);
1621         free (storeObj);
1622         *store = NULL;
1623         return;
1624    }
1625
1626
```

Oct 29 1996 16:10:17 $do.h Page 1

```
/*
 * do.h --
 *
 *      Digital Offer library header file.
 *
 *  Copyright (c) 1995 Open Market, Inc.
 *  All rights reserved.
 *
 *  This file contains proprietary and confidential information and
 *  remains the unpublished property of Open Market, Inc. Use,
 *  disclosure, or reproduction is prohibited except as permitted by
 *  express written license agreement with Open Market, Inc.
 *
 *  $Id: do.h,v 1.1.1.1 1996/07/17 21:37:00 henry Exp $
 *
 *  Henry Luo
 *  henry@openmarket.com
 */

#ifndef DO_H
#define DO_H

#ifdef __cplusplus
extern "C" {
#endif

#include "pdo.h"
#include "domsgs.h"


/*------------------------------------------------------------------------
 *
 *   Datatypes
 *
 *----------------------------------------------------------------------*/

/*
 *  Server Object
 *
 *  A server object represent a Web Server with some service.  The information
 *  encoded inside this object typically include scheme (http|https), hostname
 *  server port number, and service script (cgi) name.
 */
typedef void* OSL_Server;

/*
 *  Store Object
 *
 *  A store object contains information of all participants in the Web Store
 *  context, e.g. OM-Transact server, Softgoods fulfillment server, store's
 *  content (catalog) server, Subscription server, and the keys for the
 *  authenticated communication between the servers.
 */
typedef void* OSL_Store;

/*
 *  In Memory Cache of Keys
 */
typedef void* OSL_KeyCache;

/*  Key Object
 *
 *  A Key object contains the key used to MAC the digital offer
 */
typedef void* OSL_Key;

```

Oct 29 1996 16:10:17 do.h Page 2

```
/*------------------------------------------------------------------------
 *
 *   Function Prototypes
 *
 *----------------------------------------------------------------------*/

OSL_Status WINAPI OSL_MakeServer(OSL_Server *server, OSL_Error *error,
        OSL_Const_String scheme, OSL_Const_String host, int port,
        OSL_Const_String script);

OSL_Status WINAPI OSL_MakeStore(OSL_Store *store, OSL_Error *error,
        OSL_Const_String storeID, OSL_Server transactServer,
        OSL_Server fulfillmentServer, OSL_Server contentServer,
        OSL_Server subscriptionServer, OSL_Key key);

void WINAPI OSL_FreeServer(OSL_Server *server);

void WINAPI OSL_FreeStore(OSL_Store *store);

OSL_Status WINAPI OSL_LoadKeyCacheFromFile(OSL_KeyCache *keyCache,
        OSL_Error *error, OSL_Const_String keyfile);

void WINAPI OSL_FreeKeyCache(OSL_KeyCache *keyCache);

OSL_Status WINAPI OSL_GetKeyFromCache(OSL_Key *key, OSL_Error *error,
        OSL_Const_String storeID, OSL_KeyCache keyCache);

OSL_Status WINAPI OSL_GetKeyFromFile(OSL_Key *key, OSL_Error *error,
        OSL_Const_String storeID, OSL_Const_String keyFile);

void WINAPI OSL_FreeKey(OSL_Key *key);

OSL_Status WINAPI OSL_WriteOfferToURL(OSL_Offer offer, OSL_Error *error,
        OSL_Store store, OSL_String URLbuf, int buflen);


#ifdef __cplusplus
}
#endif

#endif /* DO_H */

```

```
/*
 * doint.h --
 *
 *      Digital Offer internal library header file.
 *
 *  Copyright (c) 1995 Open Market, Inc.
 *  All rights reserved.
 *
 *  This file contains proprietary and confidential information and
 *  remains the unpublished property of Open Market, Inc. Use,
 *  disclosure, or reproduction is prohibited except as permitted by
 *  express written license agreement with Open Market, Inc.
 *
 *  $Id: doint.h,v 1.6 1996/08/08 16:26:48 henry Exp $
 *
 *  Henry Luo
 *  henry@openmarket.com
 */

#ifndef DOINT_H
#define DOINT_H

#ifdef __cplusplus
extern "C" {
#endif

#ifdef __NETWARE__
#include "config.h.netware"
#include "nwoslmem.h"
#else
#include <config.h>
#endif  /* __NETWARE__ */

#include <stdio.h>
#include <string.h>
#include <time.h>

#include <tcl.h>
#include <omikdb.h>

#include "message.h"
#include "do.h"


typedef struct  OSL_ServerStruct {
    char *scheme;
    char *host;
    int port;
    char *script;
} OSL_ServerStruct;

typedef struct OSL_KeyStruct {
    char *kid;
    char *key;
    char *signingScheme;
    int begin;
    int end;
    int expires;
} OSL_KeyStruct;

typedef struct OSL_StoreStruct {
    char *storeID;
    OSL_ServerStruct *transactServer;
    OSL_ServerStruct *fulfillmentServer;
    OSL_ServerStruct *contentServer;
    OSL_ServerStruct *subscriptionServer;
    OSL_KeyStruct *key;
} OSL_StoreStruct;
```



```
/*
 *  Tcl has a nice dynamic string library, but we want to insulate ourselves
 *  from the library names (we might not always be linked with Tcl, and we
 *  may want to implement our own dynamic string library in the future.)
 */

#define DString                 Tcl_DString
#define DStringAppend           Tcl_DStringAppend
#define DStringTrunc            Tcl_DStringTrunc
#define DStringValue            Tcl_DStringValue
#define DStringFree             Tcl_DStringFree
#define DStringLength           Tcl_DStringLength
#define DStringInit             Tcl_DStringInit
#define DStringAppendElement    Tcl_DStringAppendElement
#define DStringStartSublist     Tcl_DStringStartSublist
#define DStringEndSublist       Tcl_DStringEndSublist


#define HashTable               Tcl_HashTable
#define HashEntry               Tcl_HashEntry
#define HashSearch              Tcl_HashSearch
#define InitHashTable           Tcl_InitHashTable
#define DeleteHashTable         Tcl_DeleteHashTable
#define CreateHashEntry         Tcl_CreateHashEntry
#define FindHashEntry           Tcl_FindHashEntry
#define DeleteHashEntry         Tcl_DeleteHashEntry
#define GetHashValue            Tcl_GetHashValue
#define SetHashValue            Tcl_SetHashValue
#define GetHashKey              Tcl_GetHashKey
#define FirstHashEntry          Tcl_FirstHashEntry
#define NextHashEntry           Tcl_NextHashEntry
#define HashStats               Tcl_HashStats

/* --- Constants --- */
#define OK 0

#define OSL_CONFFMT_FILE 2000
#define OSL_CONFFMT_DIS 2001

#define OSL_MAX_HASH_KEY_NAME  100
#define OSL_MAX_BUF_LEN  8096

#define OSL_PATCH_OFFER

/* --- define datatypes --- */
/* list is white space seperated char string */
typedef char * list_t;

/* associative array is the pointer to the hash table */
typedef HashTable *OM_aa;

/*
 *   functions in aa.c
 */
OM_aa WINAPI      OM_aaCreate(void);
void   WINAPI     OM_aaDelete(OM_aa aa);
int    WINAPI     OM_aaCopy(OM_aa dst, OM_aa src);
int    WINAPI     OM_aaAddEntry(OM_aa aa, char *key, ClientData value);
void   WINAPI     OM_aaDeleteEntry(OM_aa aa, char *key);
ClientData  WINAPI OM_aaGetEntry(OM_aa aa, char *key);
int        WINAPI  OM_aaSetEntry(OM_aa aa, char *key, ClientData value);
HashEntry * WINAPI OM_aaFirstEntry(OM_aa aa, HashSearch *searchPtr, char *key,
                                   ClientData *value);
HashEntry * WINAPI OM_aaNextEntry(OM_aa aa, HashSearch *searchPtr, char *key,
                                  ClientData *value);

/*
 *  Internal function prototypes
 */
OSL_Status WINAPI OSL_GetKeyFromKeyCache (OSL_Key *key,
```

```
141                 OSL_Error *error, OSL_Const_String keyType,
142                 OSL_Const_String storeID, int storeKeyID,
143                 time_t when, OSL_KeyCache keyCache);
144 static int OSL_CopyServer(OSL_Server *dstServer, OSL_Error *error,
145         OSL_Server srcServer);
146 static int OSL_CopyKey(OSL_Key *dstKey, OSL_Error *error, OSL_Key srcKey);
147 static void OSL_SetIfEmpty(char **dst, char *src);
148
149 static int OSL_urlEscape(char *inBuf, char *outBuf, int outBufLen,OSL_Error *e
    );
150 static int OSL_md5hash(char *src, char *outBuf,int outBufLen, OSL_Error *e);
151 static int OSL_sign(char *content, char *key, char *ss, char *outBuf,
152         int outBufLen, OSL_Error *e);
153 static int OSL_IsAbsoluteUrl(char *url);
154 int WINAPI OSL_mkpayload(char *kid, char *key, char *ss, OM_aa array, char *ou
    tBuf,
155         int outBufLen, OSL_Error *e);
156 int WINAPI OSL_urlUnparse(OM_aa array, char *outBuf,int outBufLen,OSL_Error *e
    );
157
158
159 #ifdef __cplusplus
160 }
161 #endif
162
163 #endif /* DOINT_H */
```

```
/* domsgs.h --
 *
 *      Message Codes for OSLDOMSGS Facility.
 *
 * Do *NOT* edit this file.  This file was generated from a OMI
 * message meta file by an automated utility.  Any changes made
 * directly to this file will be lost the next time the utility is
 * run.
 *
 *  Copyright (c) 1996 Open Market, Inc.
 *  All rights reserved.
 *
 *  This file contains proprietary and confidential information and
 *  remains the unpublished property of Open Market, Inc. Use,
 *  disclosure, or reproduction is prohibited except as permitted by
 *  express written license agreement with Open Market, Inc.
 *
 *  Todd M. Katz
 *  tmk@openmarket.com
 */

#ifndef DOMSGS_H
#define DOMSGS_H



#define OSLDO_VersionMajor      1       /* OSLDOMSGS meta file version - major
 number */
#define OSLDO_VersionMinor      0       /* OSLDOMSGS meta file version - minor
 number */

/* Message Mnemonic     Message Code    Message Text
 * ----------------     ------------    ------------
 */
#define OSLDO_E_CREATE_HASH_ENTRY               1       /* error - can't creat
e hash entry. %s\n */
#define OSLDO_E_ENTRY_NOT_FOUND         2       /* error - the entry %s is not
 found.\n */
#define OSLDO_E_LOAD_KEYDB              3       /* error - can't load keydb %s
\n */
#define OSLDO_E_GET_STORE_FM_KEYDB              4       /* error - store %s no
t found in key database %s\n */
#define OSLDO_E_KEY_TOO_EARLY           5       /* error - too early to use th
e key %s.%s at %s  */
#define OSLDO_E_KEY_EXPIRED             6       /* error - key %s.%s has expir
ed for validation at %s */
#define OSLDO_E_KEY_NOT_FOUND           7       /* error - no valid key found
 for store %s(%d) at %s */
#define OSLDO_E_CONTENT_HOST_NOT_SET            8       /* error - content ser
ver host not configured while using relative url for curl.\n */
#define OSLDO_E_FULFILLMENT_HOST_NOT_SET                9       /* error - ful
fillment server host not configured while using relative url for fulfillment.\n
 */
#define OSLDO_E_BUF_OVERFLOW            10      /* error - output buffer overf
low\n */
#define OSLDO_E_ALLOC_MEM               11      /* error - can't allocate memo
ry \n */
#define OSLDO_E_NULL_INPUT_BUF          12      /* error - null input buffer p
assed in\n */
#define OSLDO_E_UNKNOWN_SS              13      /* error - unknown signature s
cheme %s\n */
#define OSLDO_E_WRONG_SCHEME            14      /* error - %s is not an accept
ed scheme\n */
#define OSLDO_E_EXPIRES_BEFORE_NOW              15      /* error - offer expir
es before the current time. \n */
#define OSLDO_E_INVALID_PORT            16      /* error - %d is not a valid p
ort number. \n */
#define OSLDO_E_STOREID_KID_NOMATCH             17      /* error - StoreID %s
 does not match with kid %s. \n */
```



```
#define OSLDO_MAX_MSG           17      /* OSLDOMSGS Maximum Message Number */


#endif /* DOMSGS_H */
```

WO 98/19224 PCT/US97/19391

Oct 29 1996 16:10:50 md5.h Page 1

```
1   /* MD5.H - header file for MD5C.C
2    */
3
4   /* Copyright (C) 1991-2, RSA Data Security, Inc. Created 1991. All
5   rights reserved.
6
7   License to copy and use this software is granted provided that it
8   is identified as the "RSA Data Security, Inc. MD5 Message-Digest
9   Algorithm" in all material mentioning or referencing this software
10  or this function.
11
12  License is also granted to make and use derivative works provided
13  that such works are identified as "derived from the RSA Data
14  Security, Inc. MD5 Message-Digest Algorithm" in all material
15  mentioning or referencing the derived work.
16
17  RSA Data Security, Inc. makes no representations concerning either
18  the merchantability of this software or the suitability of this
19  software for any particular purpose. It is provided "as is"
20  without express or implied warranty of any kind.
21
22  These notices must be retained in any copies of any part of this
23  documentation and/or software.
24   */
25
26  /* MD5 context. */
27  typedef struct {
28    UINT4 state[4];                                   /* state (ABCD) */
29    UINT4 count[2];        /* number of bits, modulo 2^64 (lsb first) */
30    unsigned char buffer[64];                         /* input buffer */
31  } MD5_CTX;
32
33  void OM_MD5Init PROTO_LIST ((MD5_CTX *));
34  void OM_MD5Update PROTO_LIST
35    ((MD5_CTX *, unsigned char *, unsigned int));
36  void OM_MD5Final PROTO_LIST ((unsigned char [16], MD5_CTX *));
```

Oct 29 1996 16:10:49 md5c.c Page 1

```
/*
 *      $Id: md5c.c,v 1.2 1996/10/21 20:15:44 henry Exp $
 *
 *      MD5C.C - RSA Data Security, Inc., MD5 message-digest algorithm
 *
 *      History: $Log: md5c.c,v $
 *      History: Revision 1.2  1996/10/21 20:15:44  henry
 *      History: added @(#) sequence
 *      History: allowed NULL for fulfillment server and subscription server.
 *      History:
 *      History: Revision 1.1.1.1  1996/07/17 21:36:52  henry
 *      History: Combined libpdo and libdo modules.  Starting from V2.0.1 .
 *      History:
 *      History: Revision 1.1  1996/02/12 19:22:25  henry
 *      History: Added signing routines.
 *      History:
 *      History: Revision 1.5  1995/08/04 22:17:47  perry
 *      History: Misc fixes for version stamping
 *      History:
 *      History: Revision 1.4  1995/08/04 22:09:36  perry
 *      History: fix typo in rcsid
 *      History:
 *      History: Revision 1.3  1995/08/04 22:07:27  perry
 *      History: Add time and history stamp
 *      History:
 */
#ifndef lint
static char rcsid[] = "$Id: md5c.c,v 1.2 1996/10/21 20:15:44 henry Exp $";
#endif

/*
  MD5C.C - RSA Data Security, Inc., MD5 message-digest algorithm
Copyright (C) 1991-2, RSA Data Security, Inc. Created 1991. All
rights reserved.

License to copy and use this software is granted provided that it
is identified as the "RSA Data Security, Inc. MD5 Message-Digest
Algorithm" in all material mentioning or referencing this software
or this function.

License is also granted to make and use derivative works provided
that such works are identified as "derived from the RSA Data
Security, Inc. MD5 Message-Digest Algorithm" in all material
mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either
the merchantability of this software or the suitability of this
software for any particular purpose. It is provided "as is"
without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this
documentation and/or software.
 */

#include "global.h"
#include "md5.h"

/* Constants for OM_MD5Transform routine.
 */
#define S11 7
#define S12 12
#define S13 17
#define S14 22
#define S21 5
#define S22 9
#define S23 14
#define S24 20
#define S31 4
#define S32 11
#define S33 16
```

Oct 29 1996 16:10:49 md5c.c Page 2

```
#define S34 23
#define S41 6
#define S42 10
#define S43 15
#define S44 21

static void OM_MD5Transform PROTO_LIST ((UINT4 [4], unsigned char [64]));
static void OM_Encode PROTO_LIST
  ((unsigned char *, UINT4 *, unsigned int));
static void OM_Decode PROTO_LIST
  ((UINT4 *, unsigned char *, unsigned int));
static void OM_MD5_memcpy PROTO_LIST ((POINTER, POINTER, unsigned int));
static void OM_MD5_memset PROTO_LIST ((POINTER, int, unsigned int));

static unsigned char OM_PADDING[64] = {
  0x80, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
};

/* F, G, H and I are basic MD5 functions.
 */
#define F(x, y, z) (((x) & (y)) | ((~x) & (z)))
#define G(x, y, z) (((x) & (z)) | ((y) & (~z)))
#define H(x, y, z) ((x) ^ (y) ^ (z))
#define I(x, y, z) ((y) ^ ((x) | (~z)))

/* ROTATE_LEFT rotates x left n bits.
 */
#define ROTATE_LEFT(x, n) (((x) << (n)) | ((x) >> (32-(n))))

/* FF, GG, HH, and II transformations for rounds 1, 2, 3, and 4.
Rotation is separate from addition to prevent recomputation.
 */
#define FF(a, b, c, d, x, s, ac) { \
  (a) += F ((b), (c), (d)) + (x) + (UINT4)(ac); \
  (a) = ROTATE_LEFT ((a), (s)); \
  (a) += (b); \
  }
#define GG(a, b, c, d, x, s, ac) { \
  (a) += G ((b), (c), (d)) + (x) + (UINT4)(ac); \
  (a) = ROTATE_LEFT ((a), (s)); \
  (a) += (b); \
  }
#define HH(a, b, c, d, x, s, ac) { \
  (a) += H ((b), (c), (d)) + (x) + (UINT4)(ac); \
  (a) = ROTATE_LEFT ((a), (s)); \
  (a) += (b); \
  }
#define II(a, b, c, d, x, s, ac) { \
  (a) += I ((b), (c), (d)) + (x) + (UINT4)(ac); \
  (a) = ROTATE_LEFT ((a), (s)); \
  (a) += (b); \
  }

/* MD5 initialization. Begins an MD5 operation, writing a new context.
 */
void OM_MD5Init (context)                                        /* context */
MD5_CTX *context;
{
  context->count[0] = context->count[1] = 0;
  /* Load magic initialization constants.
*/
  context->state[0] = 0x67452301;
  context->state[1] = 0xefcdab89;
  context->state[2] = 0x98badcfe;
  context->state[3] = 0x10325476;
}

/* MD5 block update operation. Continues an MD5 message-digest
```

```
141   operation, processing another message block, and updating the
142   context.
143  */
144 void OM_MD5Update (context, input, inputLen)
                                                    /* context */
145 MD5_CTX *context;
                                                  /* input block */
146 unsigned char *input;
                                       /* length of input block */
147 unsigned int inputLen;
148 {
149   unsigned int i, index, partLen;
150
151   /* Compute number of bytes mod 64 */
152   index = (unsigned int)((context->count[0] >> 3) & 0x3F);
153
154   /* Update number of bits */
155   if ((context->count[0] += ((UINT4)inputLen << 3))
156    < ((UINT4)inputLen << 3))
157  context->count[1]++;
158   context->count[1] += ((UINT4)inputLen >> 29);
159
160   partLen = 64 - index;
161
162   /* Transform as many times as possible.
163 */
164   if (inputLen >= partLen) {
165  OM_MD5_memcpy
166    ((POINTER)&context->buffer[index], (POINTER)input, partLen);
167  OM_MD5Transform (context->state, context->buffer);
168
169  for (i = partLen; i + 63 < inputLen; i += 64)
170    OM_MD5Transform (context->state, &input[i]);
171
172  index = 0;
173   }
174   else
175  i = 0;
176
177   /* Buffer remaining input */
178   OM_MD5_memcpy
179  ((POINTER)&context->buffer[index], (POINTER)&input[i],
180   inputLen-i);
181 }
182
183 /* MD5 finalization. Ends an MD5 message-digest operation, writing the
184   the message digest and zeroizing the context.
185  */
186 void OM_MD5Final (digest, context)
                                                /* message digest */
187 unsigned char digest[16];
                                                    /* context */
188 MD5_CTX *context;
189 {
190   unsigned char bits[8];
191   unsigned int index, padLen;
192
193   /* Save number of bits */
194   OM_Encode (bits, context->count, 8);
195
196   /* Pad out to 56 mod 64.
197 */
198   index = (unsigned int)((context->count[0] >> 3) & 0x3f);
199   padLen = (index < 56) ? (56 - index) : (120 - index);
200   OM_MD5Update (context, OM_PADDING, padLen);
201
202   /* Append length (before padding) */
203   OM_MD5Update (context, bits, 8);
204   /* Store state in digest */
205   OM_Encode (digest, context->state, 16);
206
207   /* Zeroize sensitive information.
208 */
209   OM_MD5_memset ((POINTER)context, 0, sizeof (*context));
210 }
```



```
211
212 /* MD5 basic transformation. Transforms state based on block.
213  */
214 static void OM_MD5Transform (state, block)
215 UINT4 state[4];
216 unsigned char block[64];
217 {
218   UINT4 a = state[0], b = state[1], c = state[2], d = state[3], x[16];
219
220   OM_Decode (x, block, 64);
221
222   /* Round 1 */
223   FF (a, b, c, d, x[ 0], S11, 0xd76aa478); /* 1 */
224   FF (d, a, b, c, x[ 1], S12, 0xe8c7b756); /* 2 */
225   FF (c, d, a, b, x[ 2], S13, 0x242070db); /* 3 */
226   FF (b, c, d, a, x[ 3], S14, 0xc1bdceee); /* 4 */
227   FF (a, b, c, d, x[ 4], S11, 0xf57c0faf); /* 5 */
228   FF (d, a, b, c, x[ 5], S12, 0x4787c62a); /* 6 */
229   FF (c, d, a, b, x[ 6], S13, 0xa8304613); /* 7 */
230   FF (b, c, d, a, x[ 7], S14, 0xfd469501); /* 8 */
231   FF (a, b, c, d, x[ 8], S11, 0x698098d8); /* 9 */
232   FF (d, a, b, c, x[ 9], S12, 0x8b44f7af); /* 10 */
233   FF (c, d, a, b, x[10], S13, 0xffff5bb1); /* 11 */
234   FF (b, c, d, a, x[11], S14, 0x895cd7be); /* 12 */
235   FF (a, b, c, d, x[12], S11, 0x6b901122); /* 13 */
236   FF (d, a, b, c, x[13], S12, 0xfd987193); /* 14 */
237   FF (c, d, a, b, x[14], S13, 0xa679438e); /* 15 */
238   FF (b, c, d, a, x[15], S14, 0x49b40821); /* 16 */
239
240  /* Round 2 */
241   GG (a, b, c, d, x[ 1], S21, 0xf61e2562); /* 17 */
242   GG (d, a, b, c, x[ 6], S22, 0xc040b340); /* 18 */
243   GG (c, d, a, b, x[11], S23, 0x265e5a51); /* 19 */
244   GG (b, c, d, a, x[ 0], S24, 0xe9b6c7aa); /* 20 */
245   GG (a, b, c, d, x[ 5], S21, 0xd62f105d); /* 21 */
246   GG (d, a, b, c, x[10], S22,  0x2441453); /* 22 */
247   GG (c, d, a, b, x[15], S23, 0xd8a1e681); /* 23 */
248   GG (b, c, d, a, x[ 4], S24, 0xe7d3fbc8); /* 24 */
249   GG (a, b, c, d, x[ 9], S21, 0x21e1cde6); /* 25 */
250   GG (d, a, b, c, x[14], S22, 0xc33707d6); /* 26 */
251   GG (c, d, a, b, x[ 3], S23, 0xf4d50d87); /* 27 */
252   GG (b, c, d, a, x[ 8], S24, 0x455a14ed); /* 28 */
253   GG (a, b, c, d, x[13], S21, 0xa9e3e905); /* 29 */
254   GG (d, a, b, c, x[ 2], S22, 0xfcefa3f8); /* 30 */
255   GG (c, d, a, b, x[ 7], S23, 0x676f02d9); /* 31 */
256   GG (b, c, d, a, x[12], S24, 0x8d2a4c8a); /* 32 */
257
258   /* Round 3 */
259   HH (a, b, c, d, x[ 5], S31, 0xfffa3942); /* 33 */
260   HH (d, a, b, c, x[ 8], S32, 0x8771f681); /* 34 */
261   HH (c, d, a, b, x[11], S33, 0x6d9d6122); /* 35 */
262   HH (b, c, d, a, x[14], S34, 0xfde5380c); /* 36 */
263   HH (a, b, c, d, x[ 1], S31, 0xa4beea44); /* 37 */
264   HH (d, a, b, c, x[ 4], S32, 0x4bdecfa9); /* 38 */
265   HH (c, d, a, b, x[ 7], S33, 0xf6bb4b60); /* 39 */
266   HH (b, c, d, a, x[10], S34, 0xbebfbc70); /* 40 */
267   HH (a, b, c, d, x[13], S31, 0x289b7ec6); /* 41 */
268   HH (d, a, b, c, x[ 0], S32, 0xeaa127fa); /* 42 */
269   HH (c, d, a, b, x[ 3], S33, 0xd4ef3085); /* 43 */
270   HH (b, c, d, a, x[ 6], S34,  0x4881d05); /* 44 */
271   HH (a, b, c, d, x[ 9], S31, 0xd9d4d039); /* 45 */
272   HH (d, a, b, c, x[12], S32, 0xe6db99e5); /* 46 */
273   HH (c, d, a, b, x[15], S33, 0x1fa27cf8); /* 47 */
274   HH (b, c, d, a, x[ 2], S34, 0xc4ac5665); /* 48 */
275
276   /* Round 4 */
277   II (a, b, c, d, x[ 0], S41, 0xf4292244); /* 49 */
278   II (d, a, b, c, x[ 7], S42, 0x432aff97); /* 50 */
279   II (c, d, a, b, x[14], S43, 0xab9423a7); /* 51 */
280   II (b, c, d, a, x[ 5], S44, 0xfc93a039); /* 52 */
```

Oct 29 1996 16:10:49 md5c.c Page 5

```
281     II (a, b, c, d, x[12], S41, 0x655b59c3); /* 53 */
282     II (d, a, b, c, x[ 3], S42, 0x8f0ccc92); /* 54 */
283     II (c, d, a, b, x[10], S43, 0xffeff47d); /* 55 */
284     II (b, c, d, a, x[ 1], S44, 0x85845dd1); /* 56 */
285     II (a, b, c, d, x[ 8], S41, 0x6fa87e4f); /* 57 */
286     II (d, a, b, c, x[15], S42, 0xfe2ce6e0); /* 58 */
287     II (c, d, a, b, x[ 6], S43, 0xa3014314); /* 59 */
288     II (b, c, d, a, x[13], S44, 0x4e0811a1); /* 60 */
289     II (a, b, c, d, x[ 4], S41, 0xf7537e82); /* 61 */
290     II (d, a, b, c, x[11], S42, 0xbd3af235); /* 62 */
291     II (c, d, a, b, x[ 2], S43, 0x2ad7d2bb); /* 63 */
292     II (b, c, d, a, x[ 9], S44, 0xeb86d391); /* 64 */
293
294     state[0] += a;
295     state[1] += b;
296     state[2] += c;
297     state[3] += d;
298
299     /* Zeroize sensitive information.
300   */
301     OM_MD5_memset ((POINTER)x, 0, sizeof (x));
302   }
303
304   /* Encodes input (UINT4) into output (unsigned char). Assumes len is
305     a multiple of 4.
306    */
307   static void OM_Encode (output, input, len)
308   unsigned char *output;
309   UINT4 *input;
310   unsigned int len;
311   {
312     unsigned int i, j;
313
314     for (i = 0, j = 0; j < len; i++, j += 4) {
315     output[j] = (unsigned char)(input[i] & 0xff);
316     output[j+1] = (unsigned char)((input[i] >> 8) & 0xff);
317     output[j+2] = (unsigned char)((input[i] >> 16) & 0xff);
318     output[j+3] = (unsigned char)((input[i] >> 24) & 0xff);
319     }
320   }
321
322   /* Decodes input (unsigned char) into output (UINT4). Assumes len is
323     a multiple of 4.
324    */
325   static void OM_Decode (output, input, len)
326   UINT4 *output;
327   unsigned char *input;
328   unsigned int len;
329   {
330     unsigned int i, j;
331
332     for (i = 0, j = 0; j < len; i++, j += 4)
333     output[i] = ((UINT4)input[j]) | (((UINT4)input[j+1]) << 8) |
334       (((UINT4)input[j+2]) << 16) | (((UINT4)input[j+3]) << 24);
335   }
336
337   /* Note: Replace "for loop" with standard memcpy if possible.
338    */
339
340   static void OM_MD5_memcpy (output, input, len)
341   POINTER output;
342   POINTER input;
343   unsigned int len;
344   {
345     unsigned int i;
346
347     for (i = 0; i < len; i++)
348     output[i] = input[i];
349   }
350
```

Oct 29 1996 16:10:49 md5c.c Page 6

```
351   /* Note: Replace "for loop" with standard memset if possible.
352    */
353   static void OM_MD5_memset (output, value, len)
354   POINTER output;
355   int value;
356   unsigned int len;
357   {
358     unsigned int i;
359
360     for (i = 0; i < len; i++)
361     ((char *)output)[i] = (char)value;
362   }
```

Oct 29 1996 16:06:24 resource.h Page 1

```
1   //{{NO_DEPENDENCIES}}
2   // Microsoft Developer Studio generated include file.
3   // Used by coupon.rc
4   //
5   #define IDS_COUPON                      1
6   #define IDD_ABOUTBOX_COUPON             1
7   #define IDB_COUPON                      1
8   #define IDI_ABOUTDLL                    1
9   #define IDS_COUPON_PPG                  2
10  #define IDS_COUPON_PPG_CAPTION          100
11  #define IDD_PROPPAGE_COUPON             100
12  #define IDC_EDIT1                       201
13  #define IDC_EDIT2                       202
14  #define IDC_EDIT3                       203
15  #define IDC_EDIT4                       204
16
17  // Next default values for new objects
18  //
19  #ifdef APSTUDIO_INVOKED
20  #ifndef APSTUDIO_READONLY_SYMBOLS
21  #define _APS_NEXT_RESOURCE_VALUE        201
22  #define _APS_NEXT_COMMAND_VALUE         32768
23  #define _APS_NEXT_CONTROL_VALUE         205
24  #define _APS_NEXT_SYMED_VALUE           101
25  #endif
26  #endif
```

Oct 29 1996 16:42:36 omdo.cpp Page 1

```
1   // omdo.cpp : Implementation of COmdoApp and DLL registration.
2
3   #include "stdafx.h"
4   #include "omdo.h"
5
6   #ifdef _DEBUG
7   #define new DEBUG_NEW
8   #undef THIS_FILE
9   static char THIS_FILE[] = __FILE__;
10  #endif
11
12
13  COmdoApp NEAR theApp;
14
15  const GUID CDECL BASED_CODE _tlid =
16                  { 0x43b6bbc0, 0x1abb, 0x11d0, { 0xa0, 0x21, 0x44, 0x45, 0x53,
    0x54, 0, 0 } };
17  const WORD _wVerMajor = 1;
18  const WORD _wVerMinor = 0;
19
20
21  ////////////////////////////////////////////////////////////////////////////
22  // COmdoApp::InitInstance - DLL initialization
23
24  BOOL COmdoApp::InitInstance()
25  {
26          BOOL bInit = COleControlModule::InitInstance();
27
28          if (bInit)
29          {
30                  // TODO: Add your own module initialization code here.
31          }
32
33          return bInit;
34  }
35
36
37  ////////////////////////////////////////////////////////////////////////////
38  // COmdoApp::ExitInstance - DLL termination
39
40  int COmdoApp::ExitInstance()
41  {
42          // TODO: Add your own module termination code here.
43
44          return COleControlModule::ExitInstance();
45  }
46
47
48  ////////////////////////////////////////////////////////////////////////////
49  // DllRegisterServer - Adds entries to the system registry
50
51  STDAPI DllRegisterServer(void)
52  {
53          AFX_MANAGE_STATE(_afxModuleAddrThis);
54
55          if (!AfxOleRegisterTypeLib(AfxGetInstanceHandle(), _tlid))
56                  return ResultFromScode(SELFREG_E_TYPELIB);
57
58          if (!COleObjectFactoryEx::UpdateRegistryAll(TRUE))
59                  return ResultFromScode(SELFREG_E_CLASS);
60
61          return NOERROR;
62  }
63
64
65  ////////////////////////////////////////////////////////////////////////////
66  // DllUnregisterServer - Removes entries from the system registry
67
68  STDAPI DllUnregisterServer(void)
69  {
```

Oct 29 1996 16:42:36 omdo.cpp Page 2

```
70          AFX_MANAGE_STATE(_afxModuleAddrThis);
71
72          if (!AfxOleUnregisterTypeLib(_tlid))
73                  return ResultFromScode(SELFREG_E_TYPELIB);
74
75          if (!COleObjectFactoryEx::UpdateRegistryAll(FALSE))
76                  return ResultFromScode(SELFREG_E_CLASS);
77
78          return NOERROR;
79  }
```

```
1   // omdo.h : main header file for OMDO.DLL
2
3   #if !defined( __AFXCTL_H__ )
4           #error include 'afxctl.h' before including this file
5   #endif
6
7   #include "resource.h"       // main symbols
8
9   /////////////////////////////////////////////////////////////////////////////
10  // COmdoApp : See omdo.cpp for implementation.
11
12  class COmdoApp : public COleControlModule
13  {
14  public:
15          BOOL InitInstance();
16          int ExitInstance();
17  };
18
19  extern const GUID CDECL _tlid;
20  extern const WORD _wVerMajor;
21  extern const WORD _wVerMinor;
```

Oct 29 1996 16:42:37 omdoctl.cpp Page 1

```
1   // OmdoCtl.cpp : Implementation of the COmdoCtrl OLE control class.
2
3   #include "stdafx.h"
4   #include <afxinet.h>
5
6
7   #include "omdo.h"
8   #include "OmdoCtl.h"
9   #include "OmdoPpg.h"
10  #include "OmdoPropPage2.h"
11  #include <winreg.h>
12
13
14  #ifdef _DEBUG
15  #define new DEBUG_NEW
16  #undef THIS_FILE
17  static char THIS_FILE[] = __FILE__;
18  #endif
19
20  static char *radix64encode_noslash(char *in, int len);
21  static char *radix64decode_noslash(char *in, int len, int *output_len);
22  static char *common_radix64encode(unsigned char *table, char *in, int len);
23  static char *radix64encode_noslash(char *in, int len);
24  static char *commonradix64decode(unsigned char *rev_table, char *in,
25                                   int len, int *output_len);
26  static char *radix64decode_noslash(char *in, int len, int *output_len);
27
28
29  int isCreated = 0;
30
31  IMPLEMENT_DYNCREATE(COmdoCtrl, COleControl)
32
33
34  /////////////////////////////////////////////////////////////////////////////
35  // Message map
36
37  BEGIN_MESSAGE_MAP(COmdoCtrl, COleControl)
38          //{{AFX_MSG_MAP(COmdoCtrl)
39          ON_WM_LBUTTONDBLCLK()
40          ON_WM_RBUTTONDOWN()
41          //}}AFX_MSG_MAP
42          ON_OLEVERB(AFX_IDS_VERB_EDIT, OnEdit)
43          ON_OLEVERB(AFX_IDS_VERB_PROPERTIES, OnProperties)
44  END_MESSAGE_MAP()
45
46
47  /////////////////////////////////////////////////////////////////////////////
48  // Dispatch map
49
50  BEGIN_DISPATCH_MAP(COmdoCtrl, COleControl)
51          //{{AFX_DISPATCH_MAP(COmdoCtrl)
52          DISP_PROPERTY_EX(COmdoCtrl, "ProdName", GetProdName, SetProdName, VT_B
    +STR)
53          DISP_PROPERTY_EX(COmdoCtrl, "UniqueID", GetUniqueID, SetUniqueID, VT_B
    +STR)
54          DISP_PROPERTY_EX(COmdoCtrl, "Detail", GetDetail, SetDetail, VT_BSTR)
55          DISP_PROPERTY_EX(COmdoCtrl, "OfferURL", GetOfferURL, SetOfferURL, VT_B
    +STR)
56          DISP_PROPERTY_EX(COmdoCtrl, "Operation", GetOperation, SetOperation, V
    +T_BSTR)
57          DISP_PROPERTY_EX(COmdoCtrl, "Type", GetType, SetType, VT_BSTR)
58          DISP_PROPERTY_EX(COmdoCtrl, "Price", GetPrice, SetPrice, VT_BSTR)
59          DISP_PROPERTY_EX(COmdoCtrl, "Currency", GetCurrency, SetCurrency, VT_B
    +STR)
60          DISP_PROPERTY_EX(COmdoCtrl, "PdoSSI", GetPdoSSI, SetPdoSSI, VT_BSTR)
61          DISP_PROPERTY_EX(COmdoCtrl, "URL", GetURL, SetURL, VT_BSTR)
62          //}}AFX_DISPATCH_MAP
63          DISP_FUNCTION_ID(COmdoCtrl, "AboutBox", DISPID_ABOUTBOX, AboutBox, VT_
    +EMPTY, VTS_NONE)
64  END_DISPATCH_MAP()
```

Oct 29 1996 16:42:37 omdoctl.cpp Page 2

```
65
66
67  /////////////////////////////////////////////////////////////////////////////
68  // Event map
69
70  BEGIN_EVENT_MAP(COmdoCtrl, COleControl)
71          //{{AFX_EVENT_MAP(COmdoCtrl)
72          //}}AFX_EVENT_MAP
73  END_EVENT_MAP()
74
75
76  /////////////////////////////////////////////////////////////////////////////
77  // Property pages
78
79  // TODO: Add more property pages as needed.  Remember to increase the count!
80  BEGIN_PROPPAGEIDS(COmdoCtrl, 2)
81          PROPPAGEID(COmdoPropPage::guid)
82          PROPPAGEID(COmdoPropPage2::guid)
83  END_PROPPAGEIDS(COmdoCtrl)
84
85
86  /////////////////////////////////////////////////////////////////////////////
87  // Initialize class factory and guid
88
89  IMPLEMENT_OLECREATE_EX(COmdoCtrl, "OMDO.OmdoCtrl.1",
90          0x43b6bbc3, 0x1abb, 0x11d0, 0xa0, 0x21, 0x44, 0x45, 0x53, 0x54, 0, 0)
91
92
93  /////////////////////////////////////////////////////////////////////////////
94  // Type library ID and version
95
96  IMPLEMENT_OLETYPELIB(COmdoCtrl, _tlid, _wVerMajor, _wVerMinor)
97
98
99  /////////////////////////////////////////////////////////////////////////////
100 // Interface IDs
101
102 const IID BASED_CODE IID_DOmdo =
103                 { 0x43b6bbc1, 0x1abb, 0x11d0, { 0xa0, 0x21, 0x44, 0x45, 0x53,
    +0x54, 0, 0 } };
104 const IID BASED_CODE IID_DOmdoEvents =
105                 { 0x43b6bbc2, 0x1abb, 0x11d0, { 0xa0, 0x21, 0x44, 0x45, 0x53,
    +0x54, 0, 0 } };
106
107
108 /////////////////////////////////////////////////////////////////////////////
109 // Control type information
110
111 static const DWORD BASED_CODE _dwOmdoOleMisc =
112         OLEMISC_ACTIVATEWHENVISIBLE |
113         OLEMISC_SETCLIENTSITEFIRST |
114         OLEMISC_INSIDEOUT |
115         OLEMISC_CANTLINKINSIDE |
116         OLEMISC_RECOMPOSEONRESIZE;
117
118 IMPLEMENT_OLECTLTYPE(COmdoCtrl, IDS_OMDO, _dwOmdoOleMisc)
119
120
121 /////////////////////////////////////////////////////////////////////////////
122 // COmdoCtrl::COmdoCtrlFactory::UpdateRegistry -
123 // Adds or removes system registry entries for COmdoCtrl
124
125 BOOL COmdoCtrl::COmdoCtrlFactory::UpdateRegistry(BOOL bRegister)
126 {
127         // TODO: Verify that your control follows apartment-model threading ru
    +les.
128         // Refer to MFC TechNote 64 for more information.
129         // If your control does not conform to the apartment-model rules, then
130         // you must modify the code below, changing the 6th parameter from
131         // afxRegInsertable | afxRegApartmentThreading to afxRegInsertable.
```

```
132
133         if (bRegister)
134                 return AfxOleRegisterControlClass(
135                         AfxGetInstanceHandle(),
136                         m_clsid,
137                         m_lpszProgID,
138                         IDS_OMDO,
139                         IDB_OMDO,
140                         afxRegInsertable | afxRegApartmentThreading,
141                         _dwOmdoOleMisc,
142                         _tlid,
143                         _wVerMajor,
144                         _wVerMinor);
145         else
146                 return AfxOleUnregisterClass(m_clsid, m_lpszProgID);
147 }
148
149
150 /////////////////////////////////////////////////////////////////////////////
151 // Licensing strings
152
153 static const TCHAR BASED_CODE _szLicFileName[] = _T("omdo.lic");
154
155 static const WCHAR BASED_CODE _szLicString[] =
156         L"Copyright (c) 1996 ";
157
158
159 /////////////////////////////////////////////////////////////////////////////
160 // COmdoCtrl::COmdoCtrlFactory::VerifyUserLicense -
161 // Checks for existence of a user license
162
163 BOOL COmdoCtrl::COmdoCtrlFactory::VerifyUserLicense()
164 {
165         return AfxVerifyLicFile(AfxGetInstanceHandle(), _szLicFileName,
166                 _szLicString);
167 }
168
169
170 /////////////////////////////////////////////////////////////////////////////
171 // COmdoCtrl::COmdoCtrlFactory::GetLicenseKey -
172 // Returns a runtime licensing key
173
174 BOOL COmdoCtrl::COmdoCtrlFactory::GetLicenseKey(DWORD dwReserved,
175         BSTR FAR* pbstrKey)
176 {
177         if (pbstrKey == NULL)
178                 return FALSE;
179
180         *pbstrKey = SysAllocString(_szLicString);
181         return (*pbstrKey != NULL);
182 }
183
184
185 /////////////////////////////////////////////////////////////////////////////
186 // COmdoCtrl::COmdoCtrl - Constructor
187
188 COmdoCtrl::COmdoCtrl()
189 {
190
191         InitializeIIDs(&IID_DOmdo, &IID_DOmdoEvents);
192
193         // TODO: Initialize your control's instance data here.
194         m_DiscountRate = 0.0;
195         m_Ticket = _T("");
196
197         m_status = 0;
198
199         m_store = NULL;
200         m_transactServer = NULL;
201         m_fulfillmentServer = NULL;
```



```
202         m_contentServer = NULL;
203         m_subscriptionServer = NULL;
204
205         m_offer = NULL;
206
207 //      m_StoreID = _T("110000");
208         m_StoreID = _T("1308");
209
210         if ( OSL_MakeServer (&m_transactServer, &m_err, _T("https"),
211                 _T("lira.openmarket.com"), 2299,  NULL) )
212         {
213                 TRACE(m_err.message);
214                 return ;
215         }
216         if ( OSL_MakeServer (&m_contentServer, &m_err, _T("http"),
217                 _T("webint.openmarket.com"), 80,  NULL) )
218         {
219                 TRACE(m_err.message);
220                 return ;
221         }
222         if ( OSL_MakeServer (&m_fulfillmentServer, &m_err, _T("http"),
223                 _T("w2-182.openmarket.com"), 80,  NULL) )
224         {
225                 TRACE(m_err.message);
226                 return ;
227         }
228         if ( OSL_MakeServer (&m_subscriptionServer, &m_err, _T("https"),
229                 _T("lira.openmarket.com"), 2299,  NULL) )
230         {
231                 TRACE(m_err.message);
232                 return ;
233         }
234
235 //      if ( OSL_LoadKeyCacheFromFile( &m_keyCache, &m_err,
236 //                      _T("c:\\omi\\slsdkpro\\conf\\demokey.kf") ) )
237         if ( OSL_LoadKeyCacheFromFile( &m_keyCache, &m_err,
238                         _T("c:\\omi\\slsdkpro\\conf\\secret.kf") ) )
239
240         {
241                 TRACE(m_err.message);
242                 return ;
243         }
244
245
246         if ( OSL_GetKeyFromCache( &m_key, &m_err, m_StoreID,
247                         m_keyCache) )
248         {
249                 TRACE(m_err.message);
250                 return ;
251         }
252
253         if ( OSL_MakeStore( &m_store, &m_err, m_StoreID,
254                 m_transactServer, m_fulfillmentServer, m_contentServer,
255                 m_subscriptionServer, m_key) )
256         {
257                 TRACE(m_err.message);
258                 return ;
259         }
260
261
262         if ( OSL_MakeOfferFromFile (&m_offer, &m_err,
263                         _T("c:\\omi\\slsdkpro\\conf\\osl.ofr")) )
264         {
265                 TRACE(m_err.message);
266                 return ;
267         }
268
269
270
271
```

```
272         m_ProdName = _T ("");
273         m_UniqueID = _T ("");
274         m_Detail = _T ("");
275         m_OfferURL = _T ("");
276         m_Type = _T ("");
277         m_Operation = _T ("");
278         m_Price = _T ("");
279         m_Currency = _T("");
280         m_PdoSSI = _T ("");
281         //mm_URL = _T("http://paydemo.openmarket.com:80/tms-ts/bin/payment.cgi
    +?66abbf3310e078c1eeec7d338c2d6c8f:ss=env&kid=110000.120001&domain=324&curl=http
    +%3a%2f%2fwww-cs.merchant.com%3a80%2f324&amt=23&goodstype=h&desc=235");
282
283
284 }
285
286
287 /////////////////////////////////////////////////////////////////////////////
288 // COmdoCtrl::~COmdoCtrl - Destructor
289
290 COmdoCtrl::~COmdoCtrl()
291 {
292         // TODO: Cleanup your control's instance data here.
293         if (m_offer)
294             OSL_FreeOffer(&m_offer);
295
296         if (m_store)
297                 OSL_FreeStore(&m_store);
298
299         if (m_transactServer)
300                 OSL_FreeServer(&m_transactServer);
301
302         // XXX more ...
303
304
305 }
306
307
308 /////////////////////////////////////////////////////////////////////////////
309 // COmdoCtrl::OnDraw - Drawing function
310
311 void COmdoCtrl::OnDraw(
312                         CDC* pdc, const CRect& rcBounds, const CRect& rcInvali
    +d)
313 {
314         HKEY hkey;
315         CString couponKey;
316         m_couponApplied = FALSE;
317
318
319         RECT x = rcBounds;
320         CString showText;
321         double price ;
322         LPCTSTR buf;
323     unsigned char dataBuf[500]; // address of data buffer
324         LONG rc;
325
326         // TODO: Replace the following code with your own drawing code.
327
328         couponKey = _T ("");
329         couponKey += "Digital Coupons\\";
330         couponKey += m_StoreID;
331         couponKey += "\\";
332         couponKey += m_UniqueID;
333
334     if ( ERROR_SUCCESS == (rc = RegOpenKeyEx(
335                         HKEY_CURRENT_USER,couponKey,
336                         0, KEY_ALL_ACCESS, &hkey )) )
337         {
338
```



```
339         DWORD keyType;  // address of buffer for value type
340
341         DWORD bufLen;           // address of data buffer size
342
343                 bufLen = 500;
344         if (ERROR_SUCCESS == (rc = RegQueryValueEx(
345                                 hkey, "Discount Rate", NULL,
346                                 &keyType, dataBuf, &bufLen)) )
347                 {
348                         dataBuf[bufLen] = 0;
349                         m_DiscountRate = atof((const char *) dataBuf);
350
351                         bufLen = 500;
352                         if (ERROR_SUCCESS == (rc = RegQueryValueEx(
353                                 hkey, "Ticket", NULL,
354                                 &keyType, dataBuf, &bufLen)) )
355
356                         {
357                                 dataBuf[bufLen] = 0;
358                                 m_Ticket = dataBuf;
359                                 m_couponApplied = TRUE;
360                         }
361                         else
362                         {
363                                 m_Ticket = "";
364                                 m_DiscountRate = 0.0;
365                                 m_couponApplied = FALSE;
366
367                         }
368                 }
369         }
370
371         if (!m_couponApplied )
372                 pdc->FillRect(rcBounds, CBrush::FromHandle((HBRUSH)GetStockObj
    +ect(GRAY_BRUSH)));
373         else
374                 pdc->FillRect(rcBounds, CBrush::FromHandle((HBRUSH)GetStockObj
    +ect(LTGRAY_BRUSH)));
375
376         pdc->Draw3dRect( &x, 0x0f0f0f0f, 0xf0f0f0f0 );
377
378         buf = m_Price;
379         price = atof (buf);
380         price = (1.0 - m_DiscountRate) * price;
381         sprintf((char *) dataBuf, "%10.2lf", price);
382
383         showText = "$   ";
384         showText += dataBuf;
385
386         pdc->DrawText ( showText,  &x, DT_SINGLELINE |DT_CENTER |DT_VCENTER);
387
388 }
389
390
391 /////////////////////////////////////////////////////////////////////////////
392 // COmdoCtrl::DoPropExchange - Persistence support
393
394 void COmdoCtrl::DoPropExchange(CPropExchange* pPX)
395 {
396         char *encBuf;
397         LPCTSTR orgBuf;
398
399         ExchangeVersion(pPX, MAKELONG(_wVerMinor, _wVerMajor));
400         COleControl::DoPropExchange(pPX);
401
402         // TODO: Call PX_ functions for each persistent custom property.
403         PX_String (pPX, _T ("ProdName"), m_ProdName, _T("") );
404         if ( pPX -> IsLoading() )
405         {
406                 SetProdName (m_ProdName );
```


WO 98/19224 PCT/US97/19391

```
407         }
408
409         PX_String (pPX, _T ("UniqueID"), m_UniqueID, _T("") );
410         if ( pPX -> IsLoading() )
411         {
412                 SetUniqueID (m_UniqueID );
413         }
414
415
416         PX_String (pPX, _T ("Detail"), m_Detail, _T("") );
417         if ( pPX -> IsLoading() )
418         {
419                 SetDetail (m_Detail );
420         }
421
422
423         PX_String (pPX, _T ("OfferURL"), m_OfferURL, _T("") );
424         if ( pPX -> IsLoading() )
425         {
426                 SetOfferURL (m_OfferURL );
427         }
428
429
430         PX_String (pPX, _T ("Type"), m_Type, _T("") );
431         if ( pPX -> IsLoading() )
432         {
433                 SetType(m_Type);
434         }
435
436         PX_String (pPX, _T ("Operation"), m_Operation, _T("") );
437         if ( pPX -> IsLoading() )
438         {
439                 SetOperation(m_Operation);
440         }
441
442         PX_String (pPX, _T ("Price"), m_Price, _T("") );
443         if ( pPX -> IsLoading() )
444         {
445                 SetPrice(m_Price);
446         }
447
448         PX_String (pPX, _T ("Currency"), m_Currency, _T("") );
449         if ( pPX -> IsLoading() )
450         {
451                 SetCurrency(m_Currency);
452         }
453
454
455         if ( (!pPX -> IsLoading () ) && isCreated == 0)
456         {
457                 OSL_SetOfferCell (m_offer,  &m_err, "Name", OSL_Column_value,
458                         m_ProdName,  0);
459                 OSL_SetOfferCell (m_offer,  &m_err, "Price", OSL_Column_value,
460                         m_Price,  0);
461                 OSL_SetOfferCell (m_offer,  &m_err, "UniqueID", OSL_Column_val
    ue,
462                         m_UniqueID,  0);
463                 OSL_SetOfferCell (m_offer,  &m_err, "Detail", OSL_Column_value
    ,
464                         m_Detail,  0);
465                 OSL_SetOfferCell (m_offer,  &m_err, "OfferURL", OSL_Column_val
    ue,
466                         m_OfferURL,  0);
467                 OSL_SetOfferCell (m_offer,  &m_err, "Type", OSL_Column_value,
468                         m_Type,  0);
469                 OSL_SetOfferCell (m_offer,  &m_err, "Operation", OSL_Column_va
    lue,
470                         m_Operation,  0);
471 //              OSL_SetOfferCell (m_offer,  &m_err, "Currency", OSL_Column_val
```



```
    ue,
472 //                      m_Currency,  0);
473
474 //      OSL_WriteOfferToSSI (offer, &err, "This is my offer",
475 //                      m_PdoSSI.GetBufferSetLength(10000), 9999);
476
477                 if (OSL_WriteOfferToURL (m_offer, &m_err, m_store,
478                         m_URL.GetBufferSetLength(1000), 999) )
479                 {
480                         m_URL = m_err.message;
481                 } else {
482                         //m_URL.ReleaseBuffer();
483                         orgBuf = m_URL;
484                         encBuf = radix64encode_noslash((char *) orgBuf, strlen
    (orgBuf));
485                         m_URL = encBuf;
486                         free( (void *) encBuf);
487                         //isCreated = 1;
488                 }
489
490                 m_URL.ReleaseBuffer();
491
492         }
493
494         PX_String (pPX, _T ("URL"), m_URL, _T("") );
495         if ( pPX -> IsLoading() )
496         {
497                 SetURL(m_URL);
498         }
499
500 }
501
502
503 /////////////////////////////////////////////////////////////////////////////
504 // COmdoCtrl::OnResetState - Reset control to default state
505
506 void COmdoCtrl::OnResetState()
507 {
508         COleControl::OnResetState();  // Resets defaults found in DoPropExchan
    ge
509
510         // TODO: Reset any other control state here.
511 }
512
513
514 /////////////////////////////////////////////////////////////////////////////
515 // COmdoCtrl::AboutBox - Display an "About" box to the user
516
517 void COmdoCtrl::AboutBox()
518 {
519         CDialog dlgAbout(IDD_ABOUTBOX_OMDO);
520         dlgAbout.DoModal();
521 }
522
523
524 /////////////////////////////////////////////////////////////////////////////
525 // COmdoCtrl message handlers
526
527 BSTR COmdoCtrl::GetProdName()
528 {
529
530         return m_ProdName.AllocSysString();
531 }
532
533 void COmdoCtrl::SetProdName(LPCTSTR lpszNewValue)
534 {
535         m_ProdName = lpszNewValue;
536
537         SetModifiedFlag();
538 }
```


WO 98/19224 -68- PCT/US97/19391

```
539
540  BSTR COmdoCtrl::GetUniqueID()
541  {
542          return m_UniqueID.AllocSysString();
543  }
544
545  void COmdoCtrl::SetUniqueID(LPCTSTR lpszNewValue)
546  {
547          m_UniqueID = lpszNewValue;
548          SetModifiedFlag();
549  }
550
551  BSTR COmdoCtrl::GetDetail()
552  {
553          return m_Detail.AllocSysString();
554  }
555
556  void COmdoCtrl::SetDetail(LPCTSTR lpszNewValue)
557  {
558          m_Detail = lpszNewValue;
559          SetModifiedFlag();
560  }
561
562  BSTR COmdoCtrl::GetOfferURL()
563  {
564          return m_OfferURL.AllocSysString();
565  }
566
567  void COmdoCtrl::SetOfferURL(LPCTSTR lpszNewValue)
568  {
569          m_OfferURL = lpszNewValue;
570          SetModifiedFlag();
571  }
572
573
574  BSTR COmdoCtrl::GetOperation()
575  {
576
577          return m_Operation.AllocSysString();
578  }
579
580  void COmdoCtrl::SetOperation(LPCTSTR lpszNewValue)
581  {
582
583          m_Operation = lpszNewValue;
584
585          SetModifiedFlag();
586  }
587
588  BSTR COmdoCtrl::GetType()
589  {
590          return m_Type.AllocSysString();
591  }
592
593  void COmdoCtrl::SetType(LPCTSTR lpszNewValue)
594  {
595          m_Type = lpszNewValue;
596
597          SetModifiedFlag();
598  }
599
600  BSTR COmdoCtrl::GetPrice()
601  {
602
603          return m_Price.AllocSysString();
604  }
605
606  void COmdoCtrl::SetPrice(LPCTSTR lpszNewValue)
607  {
608
```



```
609          m_Price = lpszNewValue;
610
611          SetModifiedFlag();
612  }
613
614  BSTR COmdoCtrl::GetCurrency()
615  {
616
617          return m_Currency.AllocSysString();
618  }
619
620  void COmdoCtrl::SetCurrency(LPCTSTR lpszNewValue)
621  {
622
623          m_Currency = lpszNewValue;
624
625          SetModifiedFlag();
626  }
627
628  BSTR COmdoCtrl::GetPdoSSI()
629  {
630
631          return m_PdoSSI.AllocSysString();
632  }
633
634  void COmdoCtrl::SetPdoSSI(LPCTSTR lpszNewValue)
635  {
636
637          m_PdoSSI = lpszNewValue;
638          SetModifiedFlag();
639  }
640
641  BSTR COmdoCtrl::GetURL()
642  {
643          return m_URL.AllocSysString();
644  }
645
646  void COmdoCtrl::SetURL(LPCTSTR lpszNewValue)
647  {
648
649          m_URL = lpszNewValue;
650          SetModifiedFlag();
651  }
652
653  void COmdoCtrl::OnRButtonDown( UINT nFlags, CPoint point )
654  {
655          char msg[200];
656
657          if ( m_couponApplied)
658          {
659                  sprintf(msg, "A %d percent discount has been applied to this i
tem!!!", (int) (m_DiscountRate * 100.0 ));
660          }
661          else
662          {
663                  sprintf(msg, "No coupon found for this item!");
664          }
665
666          MessageBoxEx(NULL, msg, "Smart Digital Offer",
667                          MB_ICONEXCLAMATION|MB_OK, MAKELANGID(LANG_ENGLISH, SUB
LANG_ENGLISH_US));
668
669  }
670
671
672  void COmdoCtrl::OnLButtonDblClk(UINT nFlags, CPoint point)
673  {
674
675          // TODO: Add your message handler code here and/or call default
676
```

```
677         char *decBuf;
678         LPCTSTR orgBuf;
679         int output_len;
680         LPCTSTR new_DO;
681
682
683         orgBuf = m_URL;
684
685         decBuf = radix64decode_noslash((char *)orgBuf, strlen(orgBuf), &output
    _len);
686
687
688         CString myDO;
689
690         myDO = decBuf;
691         myDO += "?";
692         myDO += m_Ticket;
693
694         new_DO = myDO;
695
696         output_len = strlen(new_DO);
697
698         wchar_t *wcstr=(wchar_t *)malloc(sizeof(wchar_t)*(output_len +1));
699         mbstowcs(wcstr, new_DO, output_len+1);
700
701
702         LPOLECLIENTSITE pClientSite;
703         LPOLECONTAINER ppContainer;
704
705         if (pClientSite = this->GetClientSite())
706                 if (pClientSite->GetContainer(&ppContainer)!=S_OK)
707                         ppContainer = NULL;
708
709         HRESULT foo = HlinkNavigateString((IUnknown *)(ppContainer),wcstr);
710 //      HRESULT foo = HlinkNavigateString((IUnknown *)(ppContainer),decBuf);
711         free(wcstr);
712         free( (void *) decBuf);
713
714
715
716 //      COleControl::OnLButtonDblClk(nFlags, point);
717 }
718
719
720
721
722
723
724 /* Radix-64 encoding and decoding routines.
725  * See RFC1421 for details.
726  */
727
728
729 /* This is a modified version of RADIX64, the "normal" one, has been   */
730 /* modified to replace the / and + with the * and @ chars.             */
731 /* The "modified" version is called the _noslash version's. These work */
732 /* better in URL's.                                                    */
733
734
735 /* encode tables */
736
737 static unsigned char table_noslash[64] = {
738     'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H',
739     'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P',
740     'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X',
741     'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f',
742     'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n',
743     'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
744     'w', 'x', 'y', 'z', '0', '1', '2', '3',
745 '4', '5', '6', '7', '8', '9', '@', '*'};
```



```
746
747
748
749
750  /* decode tables */
751
752
753 static unsigned char rev_table_noslash[256] = {
754     255, 255, 255, 255, 255, 255, 255, 255,
755     255, 255, 255, 255, 255, 255, 255, 255,
756     255, 255, 255, 255, 255, 255, 255, 255,
757     255, 255, 255, 255, 255, 255, 255, 255,
758     255, 255, 255, 255, 255, 255, 255, 255,
759     255, 255, 63, 255, 255, 255, 255, 255,
760     52, 53, 54, 55, 56, 57, 58, 59,
761     60, 61, 255, 255, 255, 255, 255, 255,
762     62, 0, 1, 2, 3, 4, 5, 6,
763     7, 8, 9, 10, 11, 12, 13, 14,
764     15, 16, 17, 18, 19, 20, 21, 22,
765     23, 24, 25, 255, 255, 255, 255, 255,
766     255, 26, 27, 28, 29, 30, 31, 32,
767     33, 34, 35, 36, 37, 38, 39, 40,
768     41, 42, 43, 44, 45, 46, 47, 48,
769     49, 50, 51, 255, 255, 255, 255, 255,
770     255, 255, 255, 255, 255, 255, 255, 255,
771     255, 255, 255, 255, 255, 255, 255, 255,
772     255, 255, 255, 255, 255, 255, 255, 255,
773     255, 255, 255, 255, 255, 255, 255, 255,
774     255, 255, 255, 255, 255, 255, 255, 255,
775     255, 255, 255, 255, 255, 255, 255, 255,
776     255, 255, 255, 255, 255, 255, 255, 255,
777     255, 255, 255, 255, 255, 255, 255, 255,
778     255, 255, 255, 255, 255, 255, 255, 255,
779     255, 255, 255, 255, 255, 255, 255, 255,
780     255, 255, 255, 255, 255, 255, 255, 255,
781     255, 255, 255, 255, 255, 255, 255, 255,
782     255, 255, 255, 255, 255, 255, 255, 255,
783     255, 255, 255, 255, 255, 255, 255, 255,
784     255, 255, 255, 255, 255, 255, 255, 255,
785     255, 255, 255, 255, 255, 255, 255, 255);
786
787
788
789
790
791
792 #define sextet1(p)      ((((p)[0]) >> 2) & 0x3f)
793 #define sextet2(p)      (((((p)[0]) & 0x3) << 4) | ((((p)[1]) & 0xf0) >> 4))
794 #define sextet3(p)      (((((p)[1]) & 0xf) << 2) | ((((p)[2]) >> 6) & 0x3))
795 #define sextet4(p)      ((((p)[2]) & 0x3f)
796
797
798 static char *radix64encode_noslash(char *in, int len)
799 {
800     return (common_radix64encode(table_noslash, in, len));
801 }
802
803
804
805
806 static char *common_radix64encode(unsigned char *table, char *in, int len)
807 {
808     int i;
809     char *buf, *p;
810     int buflen;
811
812     /* Check parameters. */
813     if (in == 0 || len == 0) {
814         return (NULL);
815     }
```

```
816     buflen = ((len - 1) / 3 + 1) * 4;
817     if ((buf = (char *) malloc(buflen + 1)) == NULL) {
818         return (NULL);
819     }
820     /*
821      * Encode all but the last 1-3 bytes, since the result may have to to be
822      * padded.
823      */
824
825     p = buf;
826     for (i = 0; i < len - 3; i += 3) {
827         *p++ = table[sextet1(&in[i])];
828         *p++ = table[sextet2(&in[i])];
829         *p++ = table[sextet3(&in[i])];
830         *p++ = table[sextet4(&in[i])];
831     }
832
833     /* Encode remaining bytes. */
834     switch (len - i) {
835         case 1:
836             *p++ = table[sextet1(&in[i])];
837             *p++ = table[sextet2(&in[i])];
838             *p++ = '=';
839             *p++ = '=';
840             break;
841         case 2:
842             *p++ = table[sextet1(&in[i])];
843             *p++ = table[sextet2(&in[i])];
844             *p++ = table[sextet3(&in[i])];
845             *p++ = '=';
846             break;
847         case 3:
848             *p++ = table[sextet1(&in[i])];
849             *p++ = table[sextet2(&in[i])];
850             *p++ = table[sextet3(&in[i])];
851             *p++ = table[sextet4(&in[i])];
852             break;
853         default:
854             return (NULL);
855     }
856     *p = 0;
857     return (buf);
858 }
859
860
861
862 /* Decode radix-64 into binary. */
863
864
865
866
867
868 #define octet1(p)       (((p)[0] << 2) | (((p)[1] >> 4) & 0x3))
869 #define octet2(p)       ((((p)[1] << 4) & 0xf0) | (((p)[2] >> 2) & 0xf))
870 #define octet3(p)       (((((p)[2]) & 0x3) << 6) | ((p)[3]))
871
872
873 static char *radix64decode_noslash(char *in, int len, int *output_len)
874 {
875     return (commonradix64decode(rev_table_noslash, in, len, output_len));
876 }
877
878
879 static char *commonradix64decode(unsigned char *rev_table, char *in, int len,
    int *output_len)
880 {
881     int i;
882     unsigned char datum[4];
883     unsigned char *buf, *p;
884     int buflen;
```



```
885
886     if (in == NULL || len == 0) {
887 /*      fprintf(stderr, "decode: bad parameters %x %d\n", 0, len); */
888         return (NULL);
889     }
890     /*
891      * By definition, the length of the input buffer must be a multiple of 4.
892      */
893
894     if (len % 4 != 0) {
895 /*      fprintf(stderr, "decode: input length not a multiple of 4\n"); */
896         return (NULL);
897     }
898     buflen = (len * 3) / 4;
899
900     /* Trim padding. */
901     if (in[len - 1] == '=')
902         buflen--;
903     if (in[len - 2] == '=')
904         buflen--;
905
906     if ((buf = (unsigned char *) malloc(buflen + 1)) == NULL) {
907 /*      fprintf(stderr, "decode: unable to allocate %d bytes\n", buflen); */
908         return (NULL);
909     }
910     /* Decode all but the last four bytes. */
911
912     p = buf;
913     for (i = 0; i < len - 4; i += 4) {
914         datum[0] = rev_table[in[i]];
915         datum[1] = rev_table[in[i + 1]];
916         datum[2] = rev_table[in[i + 2]];
917         datum[3] = rev_table[in[i + 3]];
918         *p++ = octet1(datum);
919         *p++ = octet2(datum);
920         *p++ = octet3(datum);
921     }
922
923     /* And the last four bytes... */
924
925     datum[0] = rev_table[in[i]];
926     datum[1] = rev_table[in[i + 1]];
927     datum[2] = rev_table[in[i + 2]];
928     datum[3] = rev_table[in[i + 3]];
929     *p++ = octet1(datum);
930     if (in[i + 2] != '=') {
931         *p++ = octet2(datum);
932         if (in[i + 3] != '=') {
933             *p++ = octet3(datum);
934         }
935     }
936     *output_len = buflen;
937     *(buf + buflen) = 0;
938     return ((char *) buf);
939 }
940
```


WO 98/19224 PCT/US97/19391

Oct 29 1996 16:42:37 **omdoctl.h** Page 1

```
1   // OmdoCtl.h : Declaration of the COmdoCtrl OLE control class.
2
3   ////////////////////////////////////////////////////////////////////////////
4   // COmdoCtrl : See OmdoCtl.cpp for implementation.
5
6   extern "C" {
7
8   #include "pdo.h"
9   #include "do.h"
10
11  }
12
13  class COmdoCtrl : public COleControl
14  {
15          DECLARE_DYNCREATE(COmdoCtrl)
16
17  // Constructor
18  public:
19          COmdoCtrl();
20
21  // Overrides
22
23          // Drawing function
24          virtual void OnDraw(
25                                  CDC* pdc, const CRect& rcBounds, const CRect&
    *rcInvalid);
26          // Persistence
27          virtual void DoPropExchange(CPropExchange* pPX);
28
29          // Reset control state
30          virtual void OnResetState();
31
32
33
34
35
36  // Implementation
37  protected:
38          ~COmdoCtrl();
39
40          BEGIN_OLEFACTORY(COmdoCtrl)         // Class factory and guid
41                  virtual BOOL VerifyUserLicense();
42                  virtual BOOL GetLicenseKey(DWORD, BSTR FAR*);
43          END_OLEFACTORY(COmdoCtrl)
44
45          DECLARE_OLETYPELIB(COmdoCtrl)       // GetTypeInfo
46          DECLARE_PROPPAGEIDS(COmdoCtrl)      // Property page IDs
47          DECLARE_OLECTLTYPE(COmdoCtrl)           // Type name and misc status
48
49  // Message maps
50          //{{AFX_MSG(COmdoCtrl)
51          afx_msg void OnLButtonDblClk(UINT nFlags, CPoint point);
52          afx_msg void OnRButtonDown(UINT nFlags, CPoint point);
53          //}}AFX_MSG
54          DECLARE_MESSAGE_MAP()
55
56  // Dispatch maps
57          //{{AFX_DISPATCH(COmdoCtrl)
58          afx_msg BSTR GetProdName();
59          afx_msg void SetProdName(LPCTSTR lpszNewValue);
60          afx_msg BSTR GetUniqueID();
61          afx_msg void SetUniqueID(LPCTSTR lpszNewValue);
62          afx_msg BSTR GetDetail();
63          afx_msg void SetDetail(LPCTSTR lpszNewValue);
64          afx_msg BSTR GetOfferURL();
65          afx_msg void SetOfferURL(LPCTSTR lpszNewValue);
66          afx_msg BSTR GetOperation();
67          afx_msg void SetOperation(LPCTSTR lpszNewValue);
68          afx_msg BSTR GetType();
69          afx_msg void SetType(LPCTSTR lpszNewValue);
```

Oct 29 1996 16:42:37 **omdoctl.h** Page 2

```
70          afx_msg BSTR GetPrice();
71          afx_msg void SetPrice(LPCTSTR lpszNewValue);
72          afx_msg BSTR GetCurrency();
73          afx_msg void SetCurrency(LPCTSTR lpszNewValue);
74          afx_msg BSTR GetPdoSSI();
75          afx_msg void SetPdoSSI(LPCTSTR lpszNewValue);
76          afx_msg BSTR GetURL();
77          afx_msg void SetURL(LPCTSTR lpszNewValue);
78          //}}AFX_DISPATCH
79          DECLARE_DISPATCH_MAP()
80
81          afx_msg void AboutBox();
82
83  // Event maps
84          //{{AFX_EVENT(COmdoCtrl)
85          //}}AFX_EVENT
86          DECLARE_EVENT_MAP()
87
88  // Dispatch and event IDs
89  public:
90          enum {
91          //{{AFX_DISP_ID(COmdoCtrl)
92          dispidProdName = 1L,
93          dispidUniqueID = 2L,
94          dispidDetail = 3L,
95          dispidOfferURL = 4L,
96          dispidOperation = 5L,
97          dispidProdType = 6L,
98          dispidPrice = 7L,
99          dispidCurrency = 8L,
100         dispidPdoSSI = 9L,
101         dispidURL = 10L,
102         //}}AFX_DISP_ID
103         };
104 private:
105         CString m_ProdName;
106         CString m_UniqueID;
107         CString m_Detail;
108         CString m_OfferURL;
109         CString m_Type;
110         CString m_Operation;
111         CString m_Price;
112         CString m_Currency;
113         CString m_PdoSSI;
114         CString m_StoreID;
115
116         CString m_URL;
117
118         OSL_Offer m_offer ;
119         OSL_Error m_err;
120         OSL_Store m_store;
121         OSL_Key   m_key;
122         OSL_KeyCache m_keyCache;
123         OSL_Server m_transactServer;
124         OSL_Server m_fulfillmentServer;
125         OSL_Server m_contentServer;
126         OSL_Server m_subscriptionServer;
127
128         int m_status;
129         double m_DiscountRate;
130         CString m_Ticket;
131         int m_couponApplied;
132
133
134
135
136 };
```

Oct 29 1996 16:42:37 omdoppg.cpp Page 1

```
1   // OmdoPpg.cpp : Implementation of the COmdoPropPage property page class.
2
3   #include "stdafx.h"
4   #include "omdo.h"
5   #include "OmdoPpg.h"
6
7   #ifdef _DEBUG
8   #define new DEBUG_NEW
9   #undef THIS_FILE
10  static char THIS_FILE[] = __FILE__;
11  #endif
12
13
14  IMPLEMENT_DYNCREATE(COmdoPropPage, COlePropertyPage)
15
16
17  ////////////////////////////////////////////////////////////////////////////
18  // Message map
19
20  BEGIN_MESSAGE_MAP(COmdoPropPage, COlePropertyPage)
21          //{{AFX_MSG_MAP(COmdoPropPage)
22          // NOTE - ClassWizard will add and remove message map entries
23          //    DO NOT EDIT what you see in these blocks of generated code !
24          //}}AFX_MSG_MAP
25  END_MESSAGE_MAP()
26
27
28  ////////////////////////////////////////////////////////////////////////////
29  // Initialize class factory and guid
30
31  IMPLEMENT_OLECREATE_EX(COmdoPropPage, "OMDO.OmdoPropPage.1",
32          0x43b6bbc4, 0x1abb, 0x11d0, 0xa0, 0x21, 0x44, 0x45, 0x53, 0x54, 0, 0)
33
34
35  ////////////////////////////////////////////////////////////////////////////
36  // COmdoPropPage::COmdoPropPageFactory::UpdateRegistry -
37  // Adds or removes system registry entries for COmdoPropPage
38
39  BOOL COmdoPropPage::COmdoPropPageFactory::UpdateRegistry(BOOL bRegister)
40  {
41          if (bRegister)
42                  return AfxOleRegisterPropertyPageClass(AfxGetInstanceHandle(),
43                          m_clsid, IDS_OMDO_PPG);
44          else
45                  return AfxOleUnregisterClass(m_clsid, NULL);
46  }
47
48
49  ////////////////////////////////////////////////////////////////////////////
50  // COmdoPropPage::COmdoPropPage - Constructor
51
52  COmdoPropPage::COmdoPropPage() :
53          COlePropertyPage(IDD, IDS_OMDO_PPG_CAPTION)
54  {
55          //{{AFX_DATA_INIT(COmdoPropPage)
56          m_ProdName = _T("");
57          m_UniqueID = _T("");
58          m_OfferURL = _T("");
59          m_Detail = _T("");
60          m_Operation = _T("");
61          m_Type = _T("");
62          //}}AFX_DATA_INIT
63  }
64
65
66  ////////////////////////////////////////////////////////////////////////////
67  // COmdoPropPage::DoDataExchange - Moves data between page and properties
68
69  void COmdoPropPage::DoDataExchange(CDataExchange* pDX)
70  {
```

Oct 29 1996 16:42:37 omdoppg.cpp Page 2

```
71          //{{AFX_DATA_MAP(COmdoPropPage)
72          DDP_Text(pDX, IDC_EDIT1, m_ProdName, _T("ProdName") );
73          DDX_Text(pDX, IDC_EDIT1, m_ProdName);
74          DDP_Text(pDX, IDC_EDIT2, m_UniqueID, _T("UniqueID") );
75          DDX_Text(pDX, IDC_EDIT2, m_UniqueID);
76          DDP_Text(pDX, IDC_EDIT3, m_OfferURL, _T("OfferURL") );
77          DDX_Text(pDX, IDC_EDIT3, m_OfferURL);
78          DDP_Text(pDX, IDC_EDIT4, m_Detail, _T("Detail") );
79          DDX_Text(pDX, IDC_EDIT4, m_Detail);
80          DDP_CBString(pDX, IDC_COMBO2, m_Operation, _T("Operation") );
81          DDX_CBString(pDX, IDC_COMBO2, m_Operation);
82          DDP_CBString(pDX, IDC_COMBO1, m_Type, _T("Type") );
83          DDX_CBString(pDX, IDC_COMBO1, m_Type);
84          //}}AFX_DATA_MAP
85          DDP_PostProcessing(pDX);
86  }
87
88  ////////////////////////////////////////////////////////////////////////////
89  // COmdoPropPage message handlers
90
```

Oct 29 1996 16:42:38 omdoppg.h Page 1

```
1   // OmdoPpg.h : Declaration of the COmdoPropPage property page class.
2
3   ////////////////////////////////////////////////////////////////////////
4   // COmdoPropPage : See OmdoPpg.cpp.cpp for implementation.
5
6   class COmdoPropPage : public COlePropertyPage
7   {
8           DECLARE_DYNCREATE(COmdoPropPage)
9           DECLARE_OLECREATE_EX(COmdoPropPage)
10
11  // Constructor
12  public:
13          COmdoPropPage();
14
15  // Dialog Data
16          //{{AFX_DATA(COmdoPropPage)
17          enum { IDD = IDD_PROPPAGE_OMDO };
18          CString m_ProdName;
19          CString m_UniqueID;
20          CString m_OfferURL;
21          CString m_Detail;
22          CString m_Operation;
23          CString m_Type;
24          //}}AFX_DATA
25
26  // Implementation
27  protected:
28          virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
29
30  // Message maps
31  protected:
32          //{{AFX_MSG(COmdoPropPage)
33                  // NOTE - ClassWizard will add and remove member functions her
    e.
34                  //    DO NOT EDIT what you see in these blocks of generated co
    de !
35          //}}AFX_MSG
36          DECLARE_MESSAGE_MAP()
37
38  };
```

Oct 29 1996 16:42:38 **omdoproppage2.cpp** Page 1

```
1   // OmdoPropPage2.cpp : Implementation file
2   //
3
4   #include "stdafx.h"
5   #include "omdo.h"
6   #include "OmdoPropPage2.h"
7
8   #ifdef _DEBUG
9   #define new DEBUG_NEW
10  #undef THIS_FILE
11  static char THIS_FILE[] = __FILE__;
12  #endif
13
14  /////////////////////////////////////////////////////////////////////////////
15  // COmdoPropPage2 dialog
16
17  IMPLEMENT_DYNCREATE(COmdoPropPage2, COlePropertyPage)
18
19
20  /////////////////////////////////////////////////////////////////////////////
21  // Message map
22
23  BEGIN_MESSAGE_MAP(COmdoPropPage2, COlePropertyPage)
24          //{{AFX_MSG_MAP(COmdoPropPage2)
25
26          //}}AFX_MSG_MAP
27  END_MESSAGE_MAP()
28
29
30  /////////////////////////////////////////////////////////////////////////////
31  // Initialize class factory and guid
32
33  // {2B3FC4A1-1AD5-11D0-A021-4445535400000}
34  IMPLEMENT_OLECREATE_EX(COmdoPropPage2, "omdo.COmdoPropPage2",
35          0x2b3fc4a1, 0x1ad5, 0x11d0, 0xa0, 0x21, 0x44, 0x45, 0x53, 0x54, 0x0, 0
    x0)
36
37
38  /////////////////////////////////////////////////////////////////////////////
39  // COmdoPropPage2::COmdoPropPage2Factory::UpdateRegistry -
40  // Adds or removes system registry entries for COmdoPropPage2
41
42  BOOL COmdoPropPage2::COmdoPropPage2Factory::UpdateRegistry(BOOL bRegister)
43  {
44          // TODO: Define string resource for page type; replace '0' below with
    ID.
45
46          if (bRegister)
47                  return AfxOleRegisterPropertyPageClass(AfxGetInstanceHandle(),
48                          m_clsid, IDS_OMDO_PPG2);
49          else
50                  return AfxOleUnregisterClass(m_clsid, NULL);
51  }
52
53
54  /////////////////////////////////////////////////////////////////////////////
55  // COmdoPropPage2::COmdoPropPage2 - Constructor
56
57  COmdoPropPage2::COmdoPropPage2() :
58          COlePropertyPage(IDD, IDS_OMDO_PPG_CAPTION2)
59  {
60          //{{AFX_DATA_INIT(COmdoPropPage2)
61          m_Currency = _T("");
62          m_Price = _T("");
63          //}}AFX_DATA_INIT
64  }
65
66
67  /////////////////////////////////////////////////////////////////////////////
68  // COmdoPropPage2::DoDataExchange - Moves data between page and properties
```

Oct 29 1996 16:42:38 **omdoproppage2.cpp** Page 2

```
69
70  void COmdoPropPage2::DoDataExchange(CDataExchange* pDX)
71  {
72          // NOTE: ClassWizard will add DDP, DDX, and DDV calls here
73          //    DO NOT EDIT what you see in these blocks of generated code !
74          //{{AFX_DATA_MAP(COmdoPropPage2)
75          DDP_CBString(pDX, IDC_COMBO1, m_Currency, _T("Currency") );
76          DDX_CBString(pDX, IDC_COMBO1, m_Currency);
77          DDP_Text(pDX, IDC_EDIT1, m_Price, _T("Price") );
78          DDX_Text(pDX, IDC_EDIT1, m_Price);
79          //}}AFX_DATA_MAP
80          DDP_PostProcessing(pDX);
81  }
82
83
84  /////////////////////////////////////////////////////////////////////////////
85  // COmdoPropPage2 message handlers
86
87  void COmdoPropPage2::OnCreatePDO()
88  {
89          // TODO: Add your control notification handler code here
90
91
92  }
```

Oct 29 1996 16:42:38 Omdoproppage2h Page 1

```
// OmdoPropPage2.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// COmdoPropPage2 : Property page dialog

class COmdoPropPage2 : public COlePropertyPage
{
        DECLARE_DYNCREATE(COmdoPropPage2)
        DECLARE_OLECREATE_EX(COmdoPropPage2)

// Constructors
public:
    COmdoPropPage2();

// Dialog Data
    //{{AFX_DATA(COmdoPropPage2)
        enum { IDD = IDD_PROPPAGE_OMDO2 };
        CString m_Currency;
        CString m_Price;
    //}}AFX_DATA

// Implementation
protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV sup

sport

// Message maps
protected:
    //{{AFX_MSG(COmdoPropPage2)
        afx_msg void OnCreatePDO();
    //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
```

14

```
1   /* pdo.h  -- Pre-Digital Offer API
2   **
3   **  Copyright (c) 1996 Open Market, Inc.
4   **  All rights reserved.
5   **
6   **  This software contains proprietary and confidential information and
7   **  remains the unpublished property of Open Market, Inc. Use, disclosure,
8   **  or reproduction is prohibited except as permitted by express written
9   **  license agreement with Open Market, Inc.
10  **
11  ** $Id: pdo.h,v 1.3 1996/08/08 15:36:06 henry Exp $
12  */
13
14  #ifndef PDO_H
15  #define PDO_H
16
17  #ifdef __cplusplus
18  extern "C"
19  {
20  #endif
21
22  /*-------------------------------------------------------------------------
23  **
24  **  Table of Contents
25  **
26  **      Definitions
27  **
28  **      Functions:
29  **          OSL_MakeOffer
30  **          OSL_FreeOffer
31  **
32  **          OSL_MakeOfferFromFile
33  **          OSL_MakeOfferFromString
34  **          OSL_LoadOfferFromSSI
35  **
36  **          OSL_WriteOfferToFile
37  **          OSL_WriteOfferToString
38  **          OSL_WriteOfferToSSI
39  **
40  **          OSL_SetOfferHeaderValue
41  **          OSL_GetOfferHeaderValue
42  **
43  **          OSL_SetOfferCell
44  **          OSL_GetOfferCell
45  **          OSL_RemoveOfferRow
46  **
47  **          OSL_GetOfferAttributes
48  **          OSL_GetOfferRequiredAttributes
49  **
50  **          OSL_CheckOffer
51  **          OSL_CheckOfferValueType
52  **          OSL_CheckOfferValueRequired
53  **          OSL_CheckOfferValueProhibited
54  **          OSL_CheckOfferValueConstraints
55  **
56  **          OSL_GetOfferMessage
57  **
58  **-------------------------------------------------------------------------
59  *-------------------------------------------------------------------------
60  *-------------------------------------------------------------------------
61  *  Offer Definitions
62  *-------------------------------------------------------------------------
63  *-------------------------------------------------------------------------
64  */
```



```
65      /* Environment Definitions */
66
67  #if defined(_WIN32) || defined(_MSC_VER)
68  #include <windows.h>
69  #else
70  #define WINAPI
71  #endif
72
73  #include <stdlib.h>
74
75      /* SecureLink Definitions */
76
77  #include "pdomsgs.h"
78
79  #ifndef OSL_MAX_MESSAGE
80  #define OSL_MAX_MESSAGE 512
81  #endif
82
83  #ifndef FALSE
84  #define FALSE 0
85  #endif
86
87  #ifndef TRUE
88  #define TRUE 1
89  #endif
90
91      /* Offer Base Types */
92
93  typedef char        OSL_Char;
94  typedef char*       OSL_String;
95  typedef const char* OSL_Const_String;
96  typedef int         OSL_Status;
97
98  /*-------------------------------------------------------------------------
99  **
100 **  Offer Structure
101 **
102 **  An OM-SecureLink SDK Offer is an opaque structure consisting of ...
103 **
104 **  A "Header" composed of structure properties.
105 **
106 **  A "Table" of Digital Offer (DO) attributes and associated information:
107 **
108 **      Each "Row" of the table represents a single attribute.
109 **
110 **      The "Columns" hold the attribute properties (e.g., name, value,
111 **      default value, constraint rules).
112 **
113 **      The intersection of a Row and a Column is a "Cell" holding
114 **      a single property.
115 **
116 **  An SDK Offer structure is constructed and modified as follows:
117 **
118 **  1.  Create an empty (null) Offer structure.
119 **  2.  Fill in the Header properties.
120 **  3.  Create a Row for each attribute, filling in the Cells one at a time.
121 **  4.  Retrieve and/or modify any Cells.
122 **  5.  Validate the Offer.
123 **  6.  Generate a PDO (string) from an Offer.
124 **  7.  Release/free the Offer and associated heap memory.
125 **
126 **  The SecureLink SDK functions perform one or more of these actions.
127 **
128 **-------------------------------------------------------------------------
129 */
130
131 typedef void* OSL_Offer;
```

Oct 29 1996 16:42:38 pdo.h Page 3

```
132
133         /* Offer Header Attributes */
134
135     typedef enum OSL_OfferHeaderAttribute
136     {
137         OSL_Header_error=0, OSL_Header_name, OSL_Header_version, OSL_Header_date,
138         OSL_Header_author, OSL_Header_generated, OSL_Header_translator,
139         OSL_Header_brand, OSL_Header_type
140     }
141     OSL_OfferHeaderAttribute;
142
143         /* Offer Columns */
144
145     typedef enum OSL_OfferColumn
146     {
147         OSL_Column_error=0, OSL_Column_name, OSL_Column_type, OSL_Column_default,
148         OSL_Column_value, OSL_Column_constraint, OSL_Column_comment, OSL_Column_ta
    g,
149         OSL_Column_required, OSL_Column_prohibited, OSL_Column_valueDefault,
150         OSL_Column_useDefault
151     }
152     OSL_OfferColumn;
153
154     /*-------------------------------------------------------------------------
    */
155     /*  Offer Error
    */
156     /*  -------------------------------------------------------------------------
    */
157
158     #ifndef _OSLERROR
159     #define _OSLERROR
160     typedef struct OSL_Error
161     {
162         int         status;     /* Returned status code of call, always bound  */
163         int         osError;
164         OSL_Char    message[OSL_MAX_MESSAGE];
165     }
166     OSL_Error;
167     #endif
168
169     /*  -------------------------------------------------------------------------
    */
170     /*  -------------------------------------------------------------------------
    */
171     /*  Offer Functions
    */
172     /*-------------------------------------------------------------------------
    */
173     /*  -------------------------------------------------------------------------
    */
174
175     /*  -------------------------------------------------------------------------
    */
176     /*  Make/Free an Offer.
    */
177     /*  -------------------------------------------------------------------------
    */
178
179     OSL_Status WINAPI OSL_MakeOffer(OSL_Offer* offer, OSL_Error* error);
180
181     void WINAPI OSL_FreeOffer(OSL_Offer* offer);
182
183     /*-------------------------------------------------------------------------
    */
184     /*  Import an Offer.
    */
185     /*-------------------------------------------------------------------------
    */
186
```

Oct 29 1996 16:42:38 pdo.h Page 4

```
187     OSL_Status WINAPI OSL_MakeOfferFromFile(OSL_Offer* offer, OSL_Error* error,
188                             OSL_Const_String pathname);
189
190     OSL_Status WINAPI OSL_MakeOfferFromString(OSL_Offer* offer, OSL_Error* error,
191                             OSL_Const_String description);
192
193     OSL_Status WINAPI OSL_LoadOfferFromSSI(OSL_Offer offer, OSL_Error* error,
194                             OSL_Const_String string,
195                             long* pdoOffset, long* pdoLength,
196                             long* textOffset, long* textLength);
197
198     /*-------------------------------------------------------------------------
    */
199     /*  Export an Offer.
    */
200     /*-------------------------------------------------------------------------
    */
201
202     OSL_Status WINAPI OSL_WriteOfferToFile(OSL_Offer offer, OSL_Error* error,
203                             OSL_Const_String pathname, const char *mode);
204
205     OSL_Status WINAPI OSL_WriteOfferToString(OSL_Offer offer, OSL_Error* error,
206                             OSL_String buffer, int maxSize);
207
208     OSL_Status WINAPI OSL_WriteOfferToSSI(OSL_Offer offer, OSL_Error* error,
209                             OSL_Const_String text, OSL_String pdo, int maxSize);
210
211     /*-------------------------------------------------------------------------
    */
212     /*  Set/Get Offer Header Values.
    */
213     /*-------------------------
    */
214
215     OSL_Status WINAPI OSL_SetOfferHeaderValue(OSL_Offer offer, OSL_Error* error,
216                             const OSL_OfferHeaderAttribute attr,
217                             OSL_Const_String value);
218
219     OSL_Status WINAPI OSL_GetOfferHeaderValue(OSL_Offer offer, OSL_Error* error,
220                             const OSL_OfferHeaderAttribute attr,
221                             OSL_String buffer, int maxSize);
222
223     /*-------------------------  ------------  ------------------------------
    */
224     /*  Set/Get Cells of a Offer Row
    */
225     /*-------------------------  ------------------------  --------  ---------
    */
226
227     OSL_Status WINAPI OSL_SetOfferCell(OSL_Offer offer, OSL_Error* error,
228                             OSL_Const_String rowName, const OSL_OfferColumn column
    ,
229                             OSL_Const_String value, const int makeNewRow);
230
231     OSL_Status WINAPI OSL_GetOfferCell(OSL_Offer offer, OSL_Error* error,
232                             OSL_Const_String rowName, const OSL_OfferColumn column
    ,
233                             OSL_String buffer, const int maxSize);
234
235     OSL_Status WINAPI OSL_RemoveOfferRow(OSL_Offer offer, OSL_Error* error,
236                             OSL_Const_String rowName);
237
238     /*-------------------------------------------------------------------------
    */
239     /*  Get Offer Attributes (i.e., Row name strings).
    */
240     /*-------------------------------------------------------------------------
    */
241
242     OSL_Status WINAPI OSL_GetOfferAttributes(OSL_Offer offer,
```

```
243                             OSL_Error* error,
244                             OSL_String allBuffers, const int maxBuffers,
245                             const int maxSize, int* rowsFound);
246
247     OSL_Status WINAPI OSL_GetOfferRequiredAttributes(OSL_Offer offer,
248                             OSL_Error* error,
249                             OSL_String allBuffers, const int maxBuffers,
250                             const int maxSize, int* rowsFound);
251
252     /*-------------------------------------------------------------------------
        */
253     /*  Validate an Offer.
        */
254     /*-------------------------------------------------------------------------
        */
255
256     OSL_Status WINAPI OSL_CheckOffer(OSL_Offer offer, OSL_Error* error);
257
258
259     OSL_Status WINAPI OSL_CheckOfferValueType(OSL_Offer offer, OSL_Error* error,
260                             OSL_Const_String rowName);
261
262     OSL_Status WINAPI OSL_CheckOfferValueRequired(OSL_Offer offer, OSL_Error* erro
        r,
263                             OSL_Const_String rowName);
264
265     OSL_Status WINAPI OSL_CheckOfferValueProhibited(OSL_Offer offer, OSL_Error* er
        ror,
266                             OSL_Const_String rowName);
267
268     OSL_Status WINAPI OSL_CheckOfferValueConstraints(OSL_Offer offer, OSL_Error* e
        rror,
269                             OSL_Const_String rowName);
270
271     /*-------------------------------------------------------------------------
        */
272     /*  Offer Messages
        */
273     /*-------------------------------------------------------------------------
        */
274
275     char* WINAPI OSL_GetOfferMessage(OSL_Status status);
276
277     #ifdef __cplusplus
278     }
279     #endif
280
281     #endif  /* PDO_H */
```

Oct 29 1996 16:42:39 **pdomsgs.h** Page 1

```
1   /*
2    * OM-SecureLink error codes.  Heavily based on Unix error codes.
3    */
4
5   #ifndef tlerrno_h
6   #define tlerrno_h
7
8
9   /* OM-SecureLink error */
10  #define OSL_NO_ERROR            0   /* No error.                          */
11  #define OSL_EWRITE_ONLY         1   /* Opening file output file as
                                           read.                              */
12                                  2   /* Buffer overflow.                   */
13  #define OSL_BUFOVER             3   /* Attribute (row) name is NULL or
14  #define OSL_ATTRLESS                   empty.                             */
15                                      4   /* Row for name doesn't exist.    */
16  #define OSL_NO_ROW              5   /* Hash table entry creation error    */
17  #define OSL_HASH_ENT            6   /* Invalid column value.              */
18  #define OSL_ROW_COLUMN          7   /* Row data is NULL.                  */
19  #define OSL_NO_ROW_DATA         8   /* Hash table is empty.               */
20  #define OSL_HASH_EMPTY          9   /* Attempting to rename slot to
21  #define OSL_ENTRY_EXIST                                    another existin
22
g slot.             */
23  #define OSL_OFFER_COL           10  /* Invalid Offer column value         */
24  /* reserved                     11     Reserved message 11.                */
25  #define OSL_SYS_ERROR           12  /* System error; see osErrno.         */
26  #define OSL_PARSE               13  /* Parsing error.                     */
27  #define OSL_TOO_MANY_OFFERS     14  /* Too many offers in cofiguration.   */
28  #define OSL_NONAME              15  /* Row doesn't have a name.           */
29  #define OSL_FALSE               16  /* False. (not required/prohibited)   */
30  #define OSL_MISSING_REQ         17  /* Required slot wasn't found         */
31  #define OSL_TCL_INVALID             18  /* Invalid attribute in TCL
32                                          command.                           */
33  #define OSL_TCL_ARGS                19  /* Wrong number of args in TCL cmd */
34  #define OSL_TCL_ODD                 20  /* List has odd number of items.   */
35  #define OSL_TCL_SPLIT               21  /* TCL SplitList error.            */
36  #define OSL_TRUE                22  /* True. (required/prohibited)        */
37  #define OSL_PASSED              23  /* Constraint/type check passed       */
38  #define OSL_FAILED              24  /* Constraint/type check failed       */
39  #define OSL_CELL_UNSET          25  /* Cell Value not set.                */
40  #define OSL_PROHIBITED          26  /* Row Prohibited.                    */
41  #define OSL_NULL_OFFER          27  /* Offer pointer pass in is NULL.     */
42  /* reserved                     28     Reserved message 28.                */
43  #define OSL_REQUIRED            29  /* Row is Required.                   */
44  #define OSL_LAST_ERROR          29  /* Last defined error value           */
45  #endif
```

Oct 29 1996 16:42:39 resource.h Page 1

```
1   //{{NO_DEPENDENCIES}}
2   // Microsoft Developer Studio generated include file.
3   // Used by omdo.rc
4   //
5   #define IDS_OMDO                    1
6   #define IDD_ABOUTBOX_OMDO           1
7   #define IDB_OMDO                    1
8   #define IDI_ABOUTDLL                2
9   #define IDS_OMDO_PPG                100
10  #define IDS_OMDO_PPG_CAPTION        100
11  #define IDD_PROPPAGE_OMDO           101
12  #define IDS_OMDO_PPG_CAPTION2       102
13  #define IDS_OMDO_PPG2               105
14  #define IDD_PROPPAGE_OMDO2          201
15  #define IDC_EDIT1                   202
16  #define IDC_EDIT2                   203
17  #define IDC_EDIT3                   204
18  #define IDC_EDIT4                   205
19  #define IDC_RADIO1                  206
20  #define IDC_RADIO2                  207
21  #define IDC_RADIO3                  208
22  #define IDC_COMBO1                  212
23  #define IDC_COMBO2                  215
24  #define IDC_LIST1
25
26  // Next default values for new objects
27  //
28  #ifdef APSTUDIO_INVOKED
29  #ifndef APSTUDIO_READONLY_SYMBOLS
30  #define _APS_NEXT_RESOURCE_VALUE    205
31  #define _APS_NEXT_COMMAND_VALUE     32768
32  #define _APS_NEXT_CONTROL_VALUE     217
33  #define _APS_NEXT_SYMED_VALUE       101
34  #endif
35  #endif
```

Oct 29 1996 16:42:39 stdafx.cpp Page 1

```
1   // stdafx.cpp : source file that includes just the standard includes
2   //  stdafx.pch will be the pre-compiled header
3   //  stdafx.obj will contain the pre-compiled type information
4
5   #include "stdafx.h"
```

Oct 29 1996 16:42:40 stdafx.h Page 1

```
1   // stdafx.h : include file for standard system include files,
2   //      or project specific include files that are used frequently,
3   //      but are changed infrequently
4
5   #define VC_EXTRALEAN            // Exclude rarely-used stuff from Windows head
    ers
6
7   #include <afxctl.h>         // MFC support for OLE Controls
8
9   // Delete the two includes below if you do not wish to use the MFC
10  //  database classes
11  #ifndef _UNICODE
12  #include <afxdb.h>                      // MFC database classes
13  #include <afxdao.h>                     // MFC DAO database classes
14  #endif //_UNICODE
```

What is claimed is:

1. A network-based system for controlled transfer of information, comprising:

a client computer;

a server computer; and

an information source computer;

the client computer, the server computer, and the information source computer being interconnected by a computer network;

the server computer being programmed to transmit to the client computer a document containing a channel object corresponding to a communication service to be provided over an information transfer channel between the information source computer and the client computer;

the client computer being programmed to activate the channel object received from the server computer, and, in response to activation of the channel object, to cause an access ticket to be stored that indicates that a user of the client computer permits the information source computer to communicate with the user over the channel;

the information source computer being programmed to transmit information to the client computer over the channel;

the client computer being programmed to receive the information from the information source computer over the channel, based on the stored access ticket.

2. The network-based system of claim 1 wherein the information source computer is the server computer.

3. The network-based system of claim 1 wherein the information source computer is distinct from the server computer.

4. The network-based system of claim 1 wherein the channel comprises a broadcast or multicast channel.

5. The network-based system of claim 4 wherein the channel further comprises a specific time period during which the information from the information source computer is to be transmitted over the broadcast or multicast channel.

6. The network-based system of claim 1 wherein the channel comprises the computer network linking the client computer and the information source computer and the information from the information source computer is received by the client computer via an asynchronous communication over the computer network.

7. The network-based system of claim 1 further comprising a notification server, the client computer being programmed to store the access ticket at the notification server, the notification server being programmed to receive the information from the information source computer over the channel based on the stored access ticket and to transmit the information to the client computer.

8. The network-based system of claim 7 wherein the notification server comprises a filtering mail gateway.

9. The network-based system of claim 1 wherein the client is programmed to store the access ticket at the client computer.

10. The network-based system of claim 1 wherein the channel object comprises identifying data specific to

the information to be provided by the information source computer.

11. The network-based system of claim 10 wherein the client computer is pre-programmed to activate the channel object if the identifying data falls within preset parameters.

12. The network-based system of claim 1 wherein the client computer is programmed to receive a request from the user to activate the channel object and to activate the channel object in response to the request.

13. The network-based system of claim 1 wherein the client computer is programmed to cause a message to be transmitted to the server computer indicating the user's interest in the information supplied by the information source computer.

14. The network-based system of claim 1 wherein the channel object comprises icon data and the client computer is programmed to display the icon data to the user.

15. The network-based system of claim 1 wherein the client computer is programmed to cause the access ticket to be stored for a limited period of time.

16. The network-based system of claim 1 wherein the information from the information source computer is encrypted and the client computer is programmed to receive a decryption key upon payment of a fee for use of the information and to decrypt the information from the information source computer using the key.

17. The network-based system of claim 1 wherein the communication service is an asynchronous communication service, and the client computer is programmed to receive the information from the information source computer asynchronously over the channel.

18. A method of controlling transfer of information in a computer network comprising a client computer, a server computer, and an information source computer, comprising the steps of:

transmitting from the server computer to the client computer a document containing a channel object corresponding to a communication service to be provided over an information transfer channel between the information source computer and the client computer;

activating the channel object received by the client computer from the server computer;

in response to activation of the channel object, causing an access ticket to be stored that indicates that a user of the client computer permits the information source computer to communicate with the user over the channel;

transmitting information from the information source computer to the client computer over the channel; and

receiving the information from the information source computer at the client computer over the channel based on the stored access ticket.

19. A network-based system for smart digital offer pricing, comprising:

a client computer; and

an offer-providing server computer;

the client computer and the offer-providing server computer being interconnected by a computer network;

the offer-providing server computer being programmed to transmit a document to the client computer comprising a smart digital offer object;

the client computer being programmed to store user-specific information at the client computer, to receive the document comprising the smart digital offer object, to activate the smart digital offer object at the client computer, which, upon activation, provides an offer to the client computer based on the stored user-specific information, and to transmit an acceptance of the offer to the offer-providing server together with an authenticator;

the offer-providing server being programmed to verify the authenticator and to cause the offer to be fulfilled based on verification of the authenticator.

20. The network-based system of claim 19 wherein the smart digital offer object is activated in a smart card on the client computer.

21. The network-based system of claim 19 wherein the smart digital offer comprises a digital signature or code to protect the smart digital offer against unauthorized tampering, and the client computer is programmed to receive the smart digital offer, to activate the smart digital offer on the client computer, and to transmit the smart digital offer back to the offer-providing server upon acceptance of the offer.

22. The network-based system of claim 19 wherein the client user-specific information comprises user profile information.

23. The network-based system of claim 22 wherein the client computer is programmed to ask the user whether the user wishes to reveal the user profile information and the client computer releases the user profile information for use by the smart digital offer only if the user authorizes release of the user profile information.

24. A method of smart digital offer pricing in a computer network comprising a client computer and an offer-providing server computer, comprising the steps of:

storing user-specific information at the client computer;

transmitting a document from the offer-providing server computer to the client computer comprising a smart digital offer object;

receiving, at the client computer, the document comprising the smart digital offer object;

activating the smart digital offer object at the client computer, which, upon activation, provides an offer to the client computer based on the stored user-specific information;

transmitting an acceptance of the offer from the client computer to the offer-providing server together with an authenticator;

verifying the authenticator at the offer-providing server; and

fulfilling the offer based on verification of the authenticator.

25. A network-based system for coupon-based smart digital offer pricing, comprising:

a client computer;

a coupon-providing server computer; and

an offer-providing server computer;

the client computer, the coupon-providing server computer, and the offer-providing server computer being interconnected by a computer network;

the coupon-providing server computer being programmed to transmit coupon information to the client computer together with an authenticator;

the client computer being programmed to receive the coupon information and the authenticator and to cause the coupon information and the authenticator to be stored;

the offer-providing server computer being programmed to transmit a document to the client computer corresponding to a smart digital offer object;

the client computer being programmed to receive the document corresponding to the smart digital offer object, to activate the smart digital offer object, which, upon activation, verifies the authenticator and provides an offer to the client computer based on the stored coupon information, and to transmit an acceptance of the offer to the offer-providing server.

26. The network-based system of claim 25 wherein the coupon-providing server computer is distinct from the offer-providing server computer.

27. The network-based system of claim 25 wherein the coupon information comprises a coupon expiration date.

28. The network-based system of claim 25 wherein the client computer is programmed to periodically remind the user of the coupon information.

29. The network-based system of claim 25 wherein the smart digital offer object is activated in the offer-

providing computer and the client computer is programmed to cause the coupon information to be transmitted to the offer-providing computer.

30. The network-based system of claim 29 wherein the coupon information comprises a code verifiable by the smart digital offer object to ensure validity of the coupon information.

31. The network-based system of claim 25 wherein the coupon information comprises a digital receipt corresponding to a purchase of a product.

32. The network-based system of claim 25 wherein the coupon-providing server is programmed to notify the offer-providing server of coupon distribution frequency.

33. The network-based system of claim 25 wherein the offer-providing server is programmed to notify the coupon-providing server of offer acceptance frequency.

34. A method of coupon-based smart digital offer pricing in a computer network comprising a client computer, a coupon-providing server computer, and an offer-providing server computer, comprising the steps of:

transmitting coupon information from the coupon-providing server computer to the client computer together with an authenticator;

receiving the coupon information and the authenticator at the client computer;

causing the coupon information and the authenticator to be stored;

transmitting a document from the coupon-providing server computer to the client computer corresponding to a smart digital offer object;

receiving, at the client computer the document corresponding to the smart digital offer object

activating the smart digital offer object, which, upon activation, verifies the authenticator and provides an offer to the client computer based on the stored coupon information; and

transmitting an acceptance of the offer from the client computer to the offer-providing server.

35. A network-based system for automatic transfer of information pertaining to a person profile of a user, comprising:

a client computer; and

a server computer;

the client computer and the server computer being interconnected by a computer network;

the server computer being programmed to transmit to the client computer a request for personal profile information pertaining to a user of the client computer;

the client computer being programmed to receive the request for personal profile information, and to activate a client avatar at the client computer that compares the request for personal profile information with a security profile of the user limiting access to personal profile information and that causes a subset of a personal profile of the user to be transmitted to the server computer based on the request for personal profile information and the security profile;

the server computer being programmed to transmit to the client computer information customized for the user based on the subset of the personal profile of the user.

36. The network-based system of claim 35 further comprising an agency computer programmed to store the

personal profile, wherein the client avatar causes an authorization message to be transmitted to the agency computer authorizing the agency computer to release the subset of the personal profile, and the agency computer is programmed to transmit the subset of the personal profile to the server computer.

37. The network-based system of claim 36 wherein the agency computer comprises a trusted mail server.

38. The network-based system of claim 35 wherein the security profile comprises a list of trusted server computers and the client avatar causes the subset of the personal profile of the user to be transmitted to the server computer if the server computer is on the list of trusted server computers.

39. The network-based system of claim 35 wherein the security profile comprises instructions to query the user before releasing certain items of personal profile information and the client avatar queries the user if the request for personal profile information pertains to one of the certain items of personal profile information and causes the one of the certain items of personal profile information to be transmitted to the server computer only if the client avatar receives a consent from the user.

40. The network-based system of claim 35 wherein the information customized for the user and transmitted by the server computer to the client computer comprises a commercial offer having user-specific terms based on the subset of the personal profile of the user.

41. The network-based system of claim 35 wherein the information customized for the user and transmitted

by the server computer to the client computer comprises a catalog having user-specific content.

42. The network-based system of claim 35 wherein the information customized for the user and transmitted by the server computer to the client computer contains a channel object corresponding to a channel for information transfer to the client computer.

43. The network-based system of claim 42 wherein the client computer is programmed to activate the channel object received from the server computer, and, in response to activation of the channel object, to store an access ticket that indicates that a user of the client computer permits information to be received over the channel, and to receive the information over the channel based on the stored access ticket.

44. The network-based system of claim 35 wherein the information customized for the user and transmitted by the server computer to the client computer is transmitted over a channel specified by a channel object transmitted by the server computer to the client computer.

45. A method for automatic transfer of information pertaining to a person profile of a user in a computer network comprising a client computer and a server computer, comprising the steps of:

transmitting from the server computer to the client computer a request for personal profile information pertaining to a user of the client computer;

receiving at the client computer the request for personal profile information;

activating a client avatar at the client computer that compares the request for personal profile information with a security profile of the user limiting access to personal profile information and that causes a subset of a personal profile of the user to be transmitted to the server computer based on the request for personal profile information and the security profile; and

transmitting from the server computer to the client computer information customized for the user based on the subset of the personal profile of the user.

46. A network-based system for metering of a user's access to linked information, comprising:

a client computer; and

a server computer;

the client computer and the server computer being interconnected by a computer network;

the server computer being programmed to transmit to the client computer a document containing an embedded link;

the client computer being programmed to activate the embedded link when at least a portion of the document is displayed, to record activation of the embedded link in a metering log, and to cause information stored in the metering log pertaining to activation of the embedded link to be transmitted to the server computer.

47. The network-based system of claim 46 further comprising an agency computer, wherein the client computer is programmed to communicate information from the metering log to the agency computer for storage and the agency computer is programmed to cause the information from the metering log to be transmitted to the server computer.

48. The network-based system of claim 47 wherein the agency computer is programmed to store billing records corresponding to the information from the metering log.

49. The network-based system of claim 46 wherein the client computer is programmed to cause the information stored in the metering log pertaining to activation of the embedded link to be transmitted immediately if the embedded link comprises an instruction to transmit it immediately.

50. The network-based system of claim 46 wherein the embedded link is structured to participate in display refresh of the document but is not structured to affect visual appearance of the document.

51. The network-based system of claim 50 wherein the client computer is programmed to record in the metering log mouse-click activity on the portion of the document corresponding to the embedded link and to allow the mouse-click activity to pass on to objects on the document other than the embedded link.

52. The network-based system of claim 46 wherein the embedded link is a link to a document other than the document containing the embedded link.

53. The network-based system of claim 46 wherein the embedded link is structured to participate in display refresh of the document and affects visual appearance of the document.

54. The network-based system of claim 53 wherein the embedded link is structured to require the client

computer to verify the presence of the metering log on the client computer before allowing the client computer to activate the embedded link.

55. The network-based system of claim 53 wherein the embedded link is structured to require the client computer to search for information stored on the client computer pertaining to authorization of the user activate the embedded link.

56. A method for metering a user's access to linked information in a computer network comprising a client computer and a server computer, comprising the steps of:

transmitting from the server computer to the client computer a document containing an embedded link;

activating the embedded link at the client computer when at least a portion of the document corresponding to the embedded link is displayed;

recording activation of the embedded link in a metering log; and

causing information stored in the metering log pertaining to activation of the embedded link to be transmitted to the server computer.

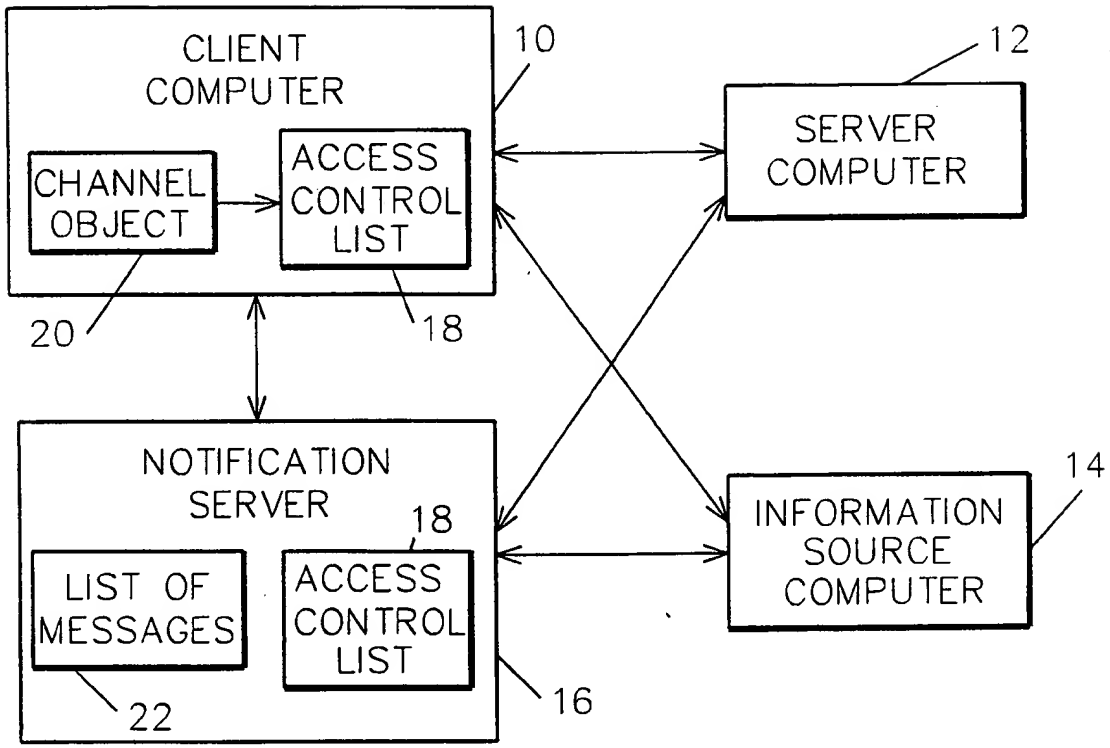
CLIENT COMPUTER
10
CHANNEL OBJECT
ACCESS CONTROL LIST
20
18
SERVER COMPUTER
12
NOTIFICATION SERVER
18
LIST OF MESSAGES
ACCESS CONTROL LIST
22
16
INFORMATION SOURCE COMPUTER
14

FIG. 1

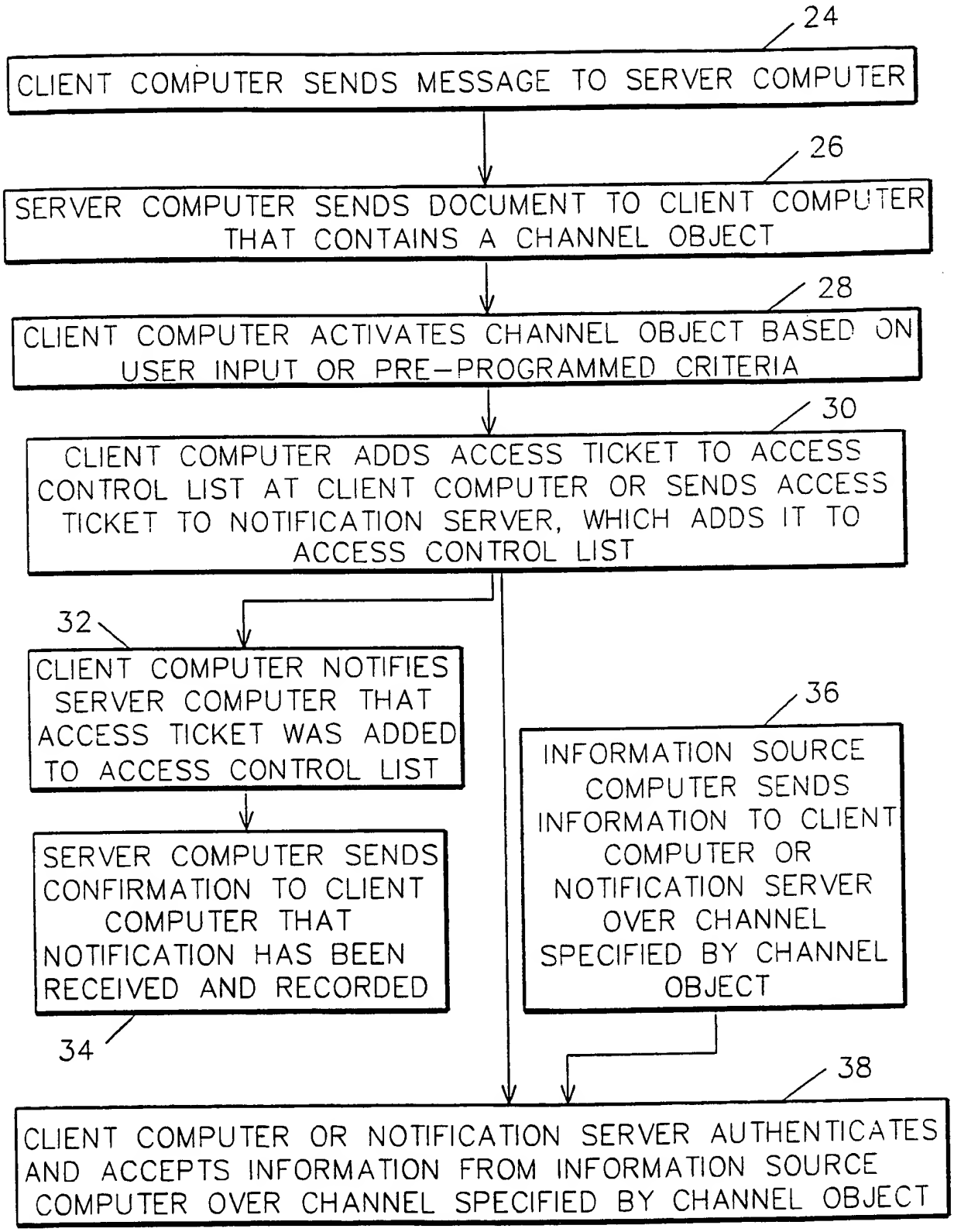
24
CLIENT COMPUTER SENDS MESSAGE TO SERVER COMPUTER
26
SERVER COMPUTER SENDS DOCUMENT TO CLIENT COMPUTER THAT CONTAINS A CHANNEL OBJECT
28
CLIENT COMPUTER ACTIVATES CHANNEL OBJECT BASED ON USER INPUT OR PRE-PROGRAMMED CRITERIA
30
CLIENT COMPUTER ADDS ACCESS TICKET TO ACCESS CONTROL LIST AT CLIENT COMPUTER OR SENDS ACCESS TICKET TO NOTIFICATION SERVER, WHICH ADDS IT TO ACCESS CONTROL LIST
32
CLIENT COMPUTER NOTIFIES SERVER COMPUTER THAT ACCESS TICKET WAS ADDED TO ACCESS CONTROL LIST
SERVER COMPUTER SENDS CONFIRMATION TO CLIENT COMPUTER THAT NOTIFICATION HAS BEEN RECEIVED AND RECORDED
34
36
INFORMATION SOURCE COMPUTER SENDS INFORMATION TO CLIENT COMPUTER OR NOTIFICATION SERVER OVER CHANNEL SPECIFIED BY CHANNEL OBJECT
38
CLIENT COMPUTER OR NOTIFICATION SERVER AUTHENTICATES AND ACCEPTS INFORMATION FROM INFORMATION SOURCE COMPUTER OVER CHANNEL SPECIFIED BY CHANNEL OBJECT

FIG. 2

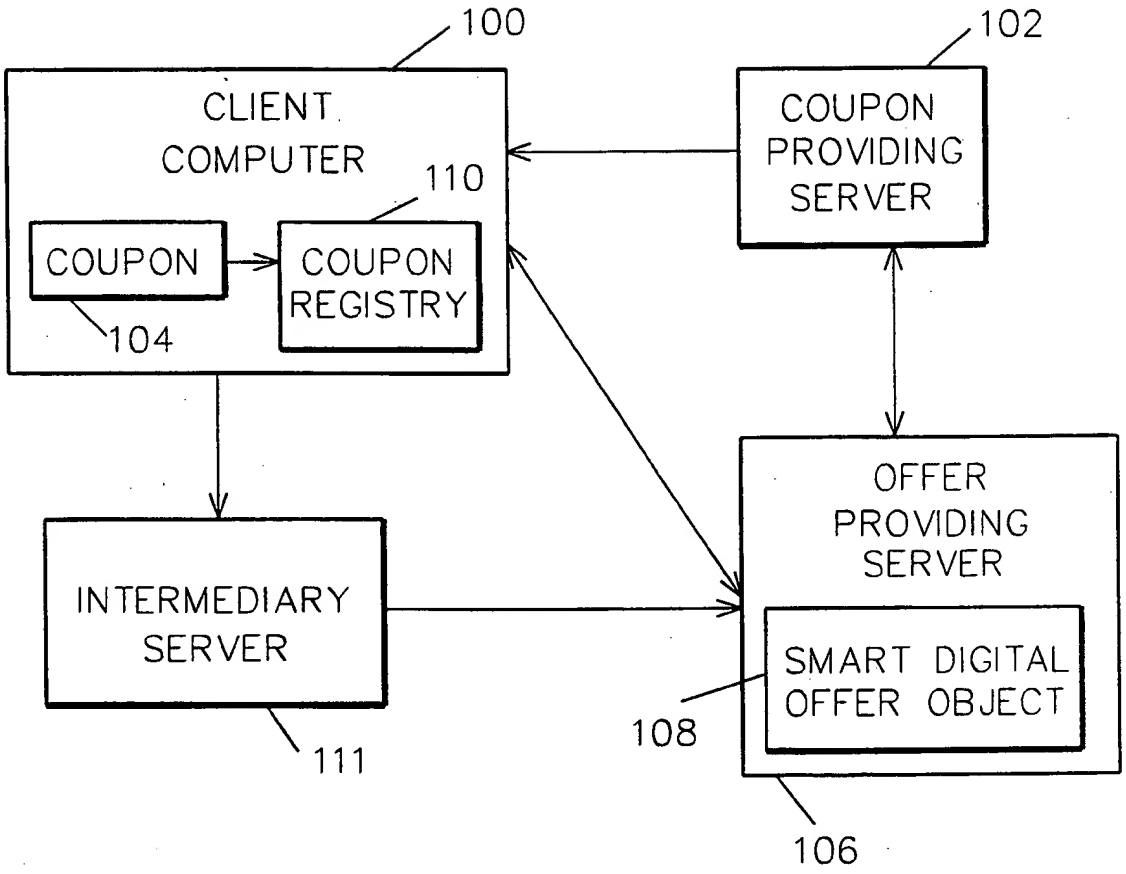
100
CLIENT COMPUTER
110
COUPON
COUPON REGISTRY
104
102
COUPON PROVIDING SERVER
OFFER PROVIDING SERVER
SMART DIGITAL OFFER OBJECT
108
106
INTERMEDIARY SERVER
111

FIG. 3

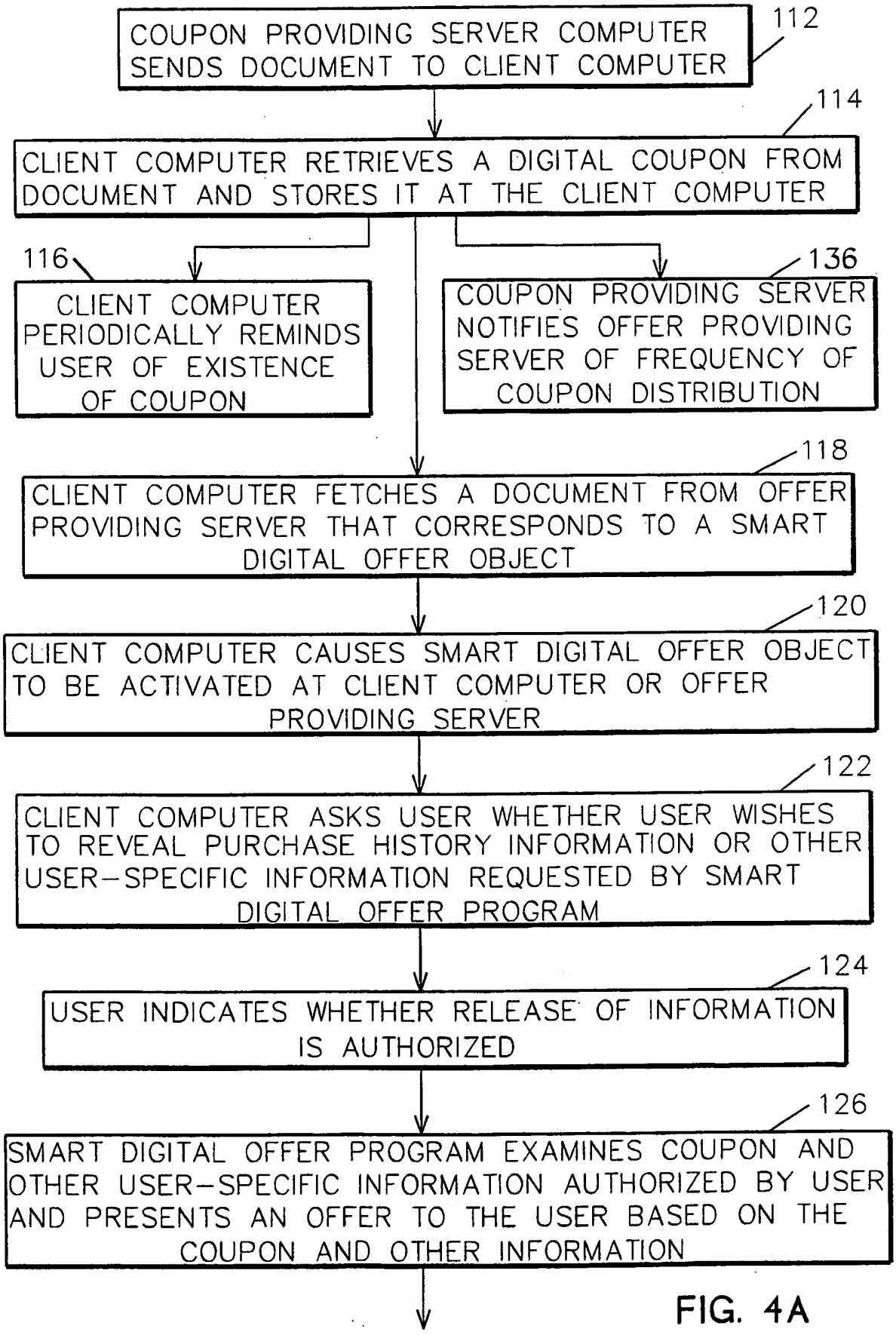
COUPON PROVIDING SERVER COMPUTER SENDS DOCUMENT TO CLIENT COMPUTER
112
CLIENT COMPUTER RETRIEVES A DIGITAL COUPON FROM DOCUMENT AND STORES IT AT THE CLIENT COMPUTER
114
116
CLIENT COMPUTER PERIODICALLY REMINDS USER OF EXISTENCE OF COUPON
136
COUPON PROVIDING SERVER NOTIFIES OFFER PROVIDING SERVER OF FREQUENCY OF COUPON DISTRIBUTION
CLIENT COMPUTER FETCHES A DOCUMENT FROM OFFER PROVIDING SERVER THAT CORRESPONDS TO A SMART DIGITAL OFFER OBJECT
118
CLIENT COMPUTER CAUSES SMART DIGITAL OFFER OBJECT TO BE ACTIVATED AT CLIENT COMPUTER OR OFFER PROVIDING SERVER
120
CLIENT COMPUTER ASKS USER WHETHER USER WISHES TO REVEAL PURCHASE HISTORY INFORMATION OR OTHER USER-SPECIFIC INFORMATION REQUESTED BY SMART DIGITAL OFFER PROGRAM
122
USER INDICATES WHETHER RELEASE OF INFORMATION IS AUTHORIZED
124
SMART DIGITAL OFFER PROGRAM EXAMINES COUPON AND OTHER USER-SPECIFIC INFORMATION AUTHORIZED BY USER AND PRESENTS AN OFFER TO THE USER BASED ON THE COUPON AND OTHER INFORMATION
126

FIG. 4A

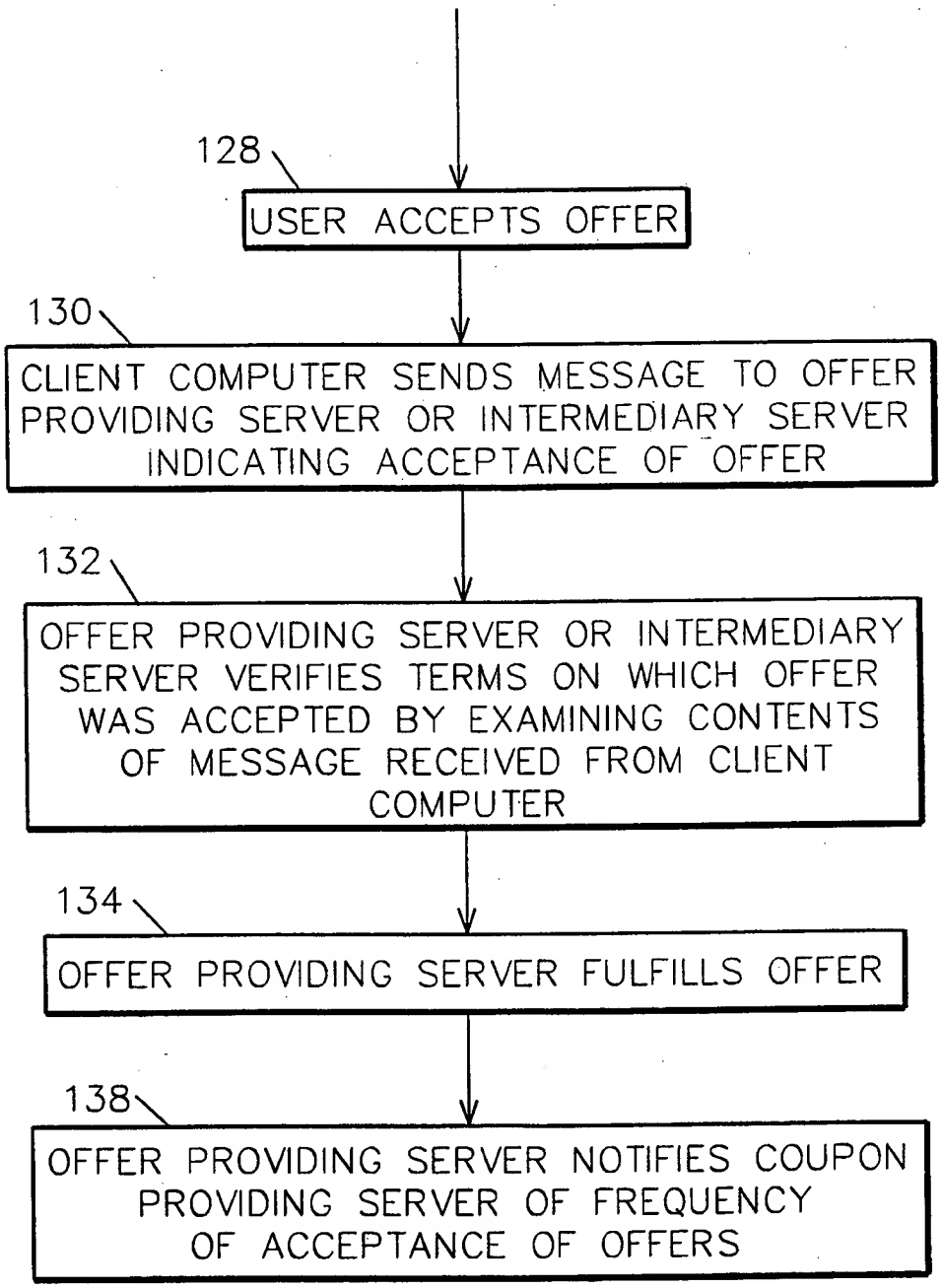
128
USER ACCEPTS OFFER
130
CLIENT COMPUTER SENDS MESSAGE TO OFFER PROVIDING SERVER OR INTERMEDIARY SERVER INDICATING ACCEPTANCE OF OFFER
132
OFFER PROVIDING SERVER OR INTERMEDIARY SERVER VERIFIES TERMS ON WHICH OFFER WAS ACCEPTED BY EXAMINING CONTENTS OF MESSAGE RECEIVED FROM CLIENT COMPUTER
134
OFFER PROVIDING SERVER FULFILLS OFFER
138
OFFER PROVIDING SERVER NOTIFIES COUPON PROVIDING SERVER OF FREQUENCY OF ACCEPTANCE OF OFFERS

FIG. 4B

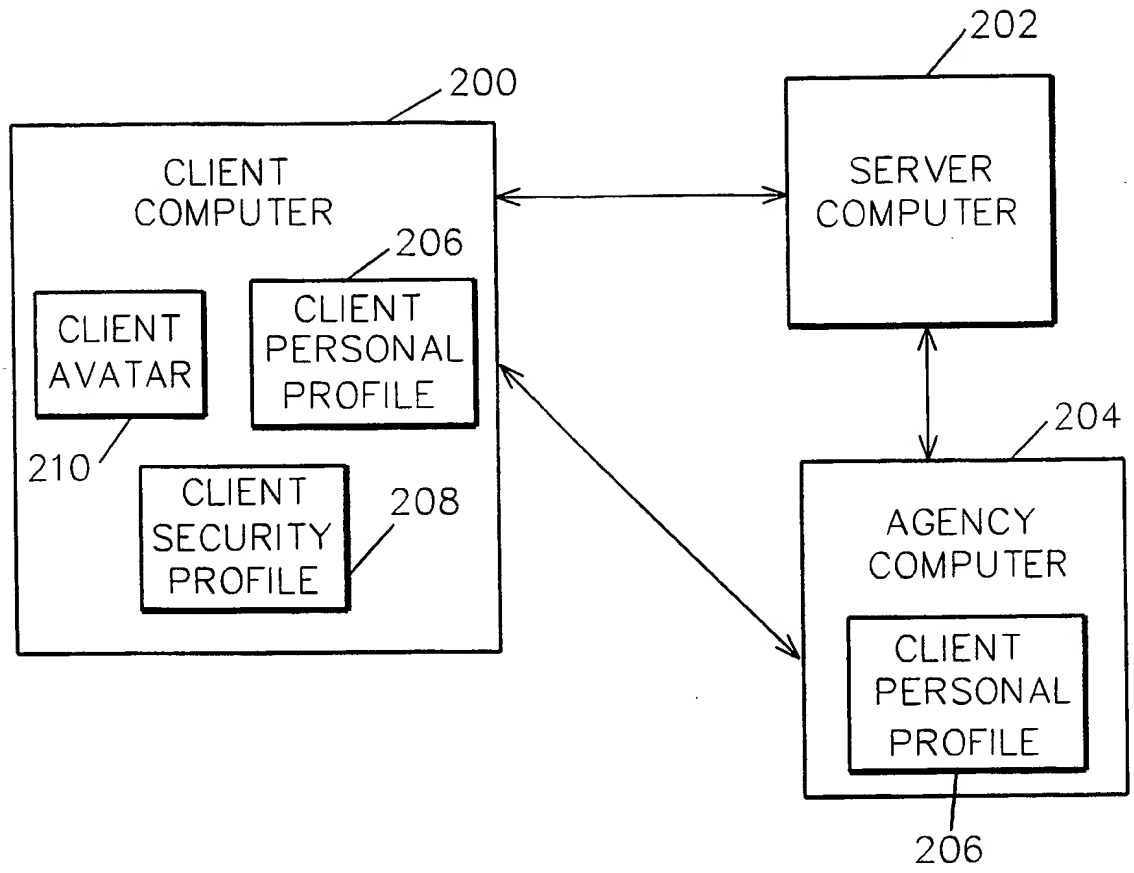
202
200
CLIENT COMPUTER
SERVER COMPUTER
206
CLIENT PERSONAL PROFILE
CLIENT AVATAR
210
CLIENT SECURITY PROFILE
208
204
AGENCY COMPUTER
CLIENT PERSONAL PROFILE
206

FIG. 5

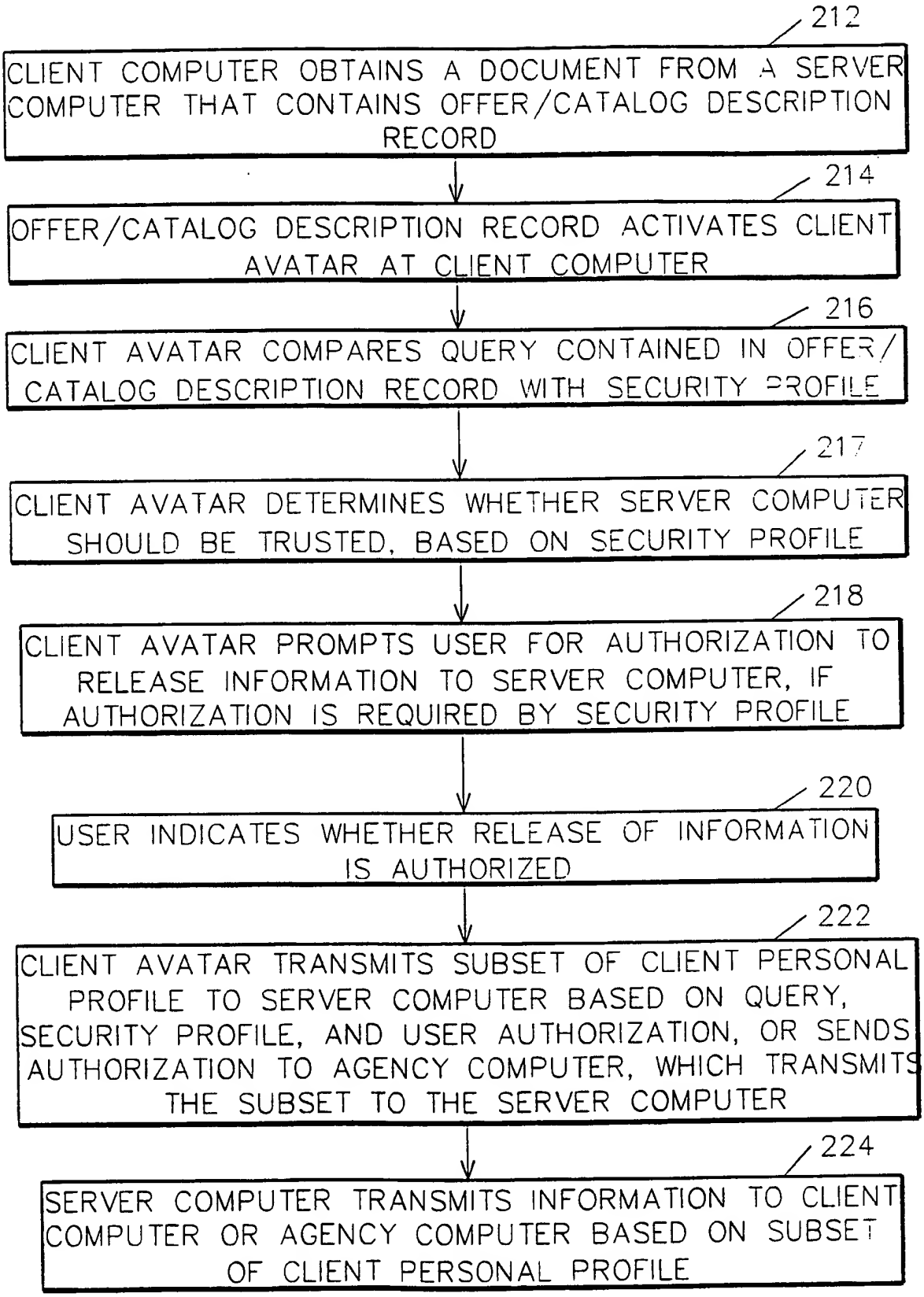
212
CLIENT COMPUTER OBTAINS A DOCUMENT FROM A SERVER COMPUTER THAT CONTAINS OFFER/CATALOG DESCRIPTION RECORD
214
OFFER/CATALOG DESCRIPTION RECORD ACTIVATES CLIENT AVATAR AT CLIENT COMPUTER
216
CLIENT AVATAR COMPARES QUERY CONTAINED IN OFFER/CATALOG DESCRIPTION RECORD WITH SECURITY PROFILE
217
CLIENT AVATAR DETERMINES WHETHER SERVER COMPUTER SHOULD BE TRUSTED, BASED ON SECURITY PROFILE
218
CLIENT AVATAR PROMPTS USER FOR AUTHORIZATION TO RELEASE INFORMATION TO SERVER COMPUTER, IF AUTHORIZATION IS REQUIRED BY SECURITY PROFILE
220
USER INDICATES WHETHER RELEASE OF INFORMATION IS AUTHORIZED
222
CLIENT AVATAR TRANSMITS SUBSET OF CLIENT PERSONAL PROFILE TO SERVER COMPUTER BASED ON QUERY, SECURITY PROFILE, AND USER AUTHORIZATION, OR SENDS AUTHORIZATION TO AGENCY COMPUTER, WHICH TRANSMITS THE SUBSET TO THE SERVER COMPUTER
224
SERVER COMPUTER TRANSMITS INFORMATION TO CLIENT COMPUTER OR AGENCY COMPUTER BASED ON SUBSET OF CLIENT PERSONAL PROFILE

FIG. 6

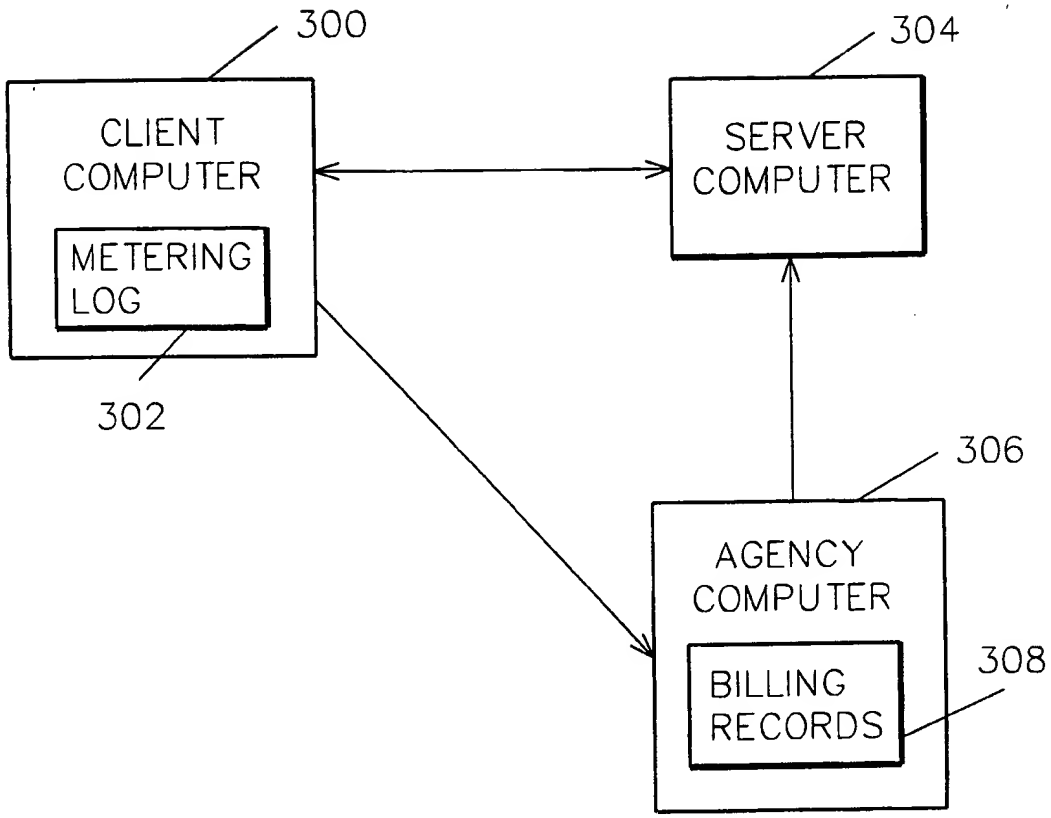
300
CLIENT COMPUTER
METERING LOG
302
304
SERVER COMPUTER
306
AGENCY COMPUTER
BILLING RECORDS
308

FIG. 7

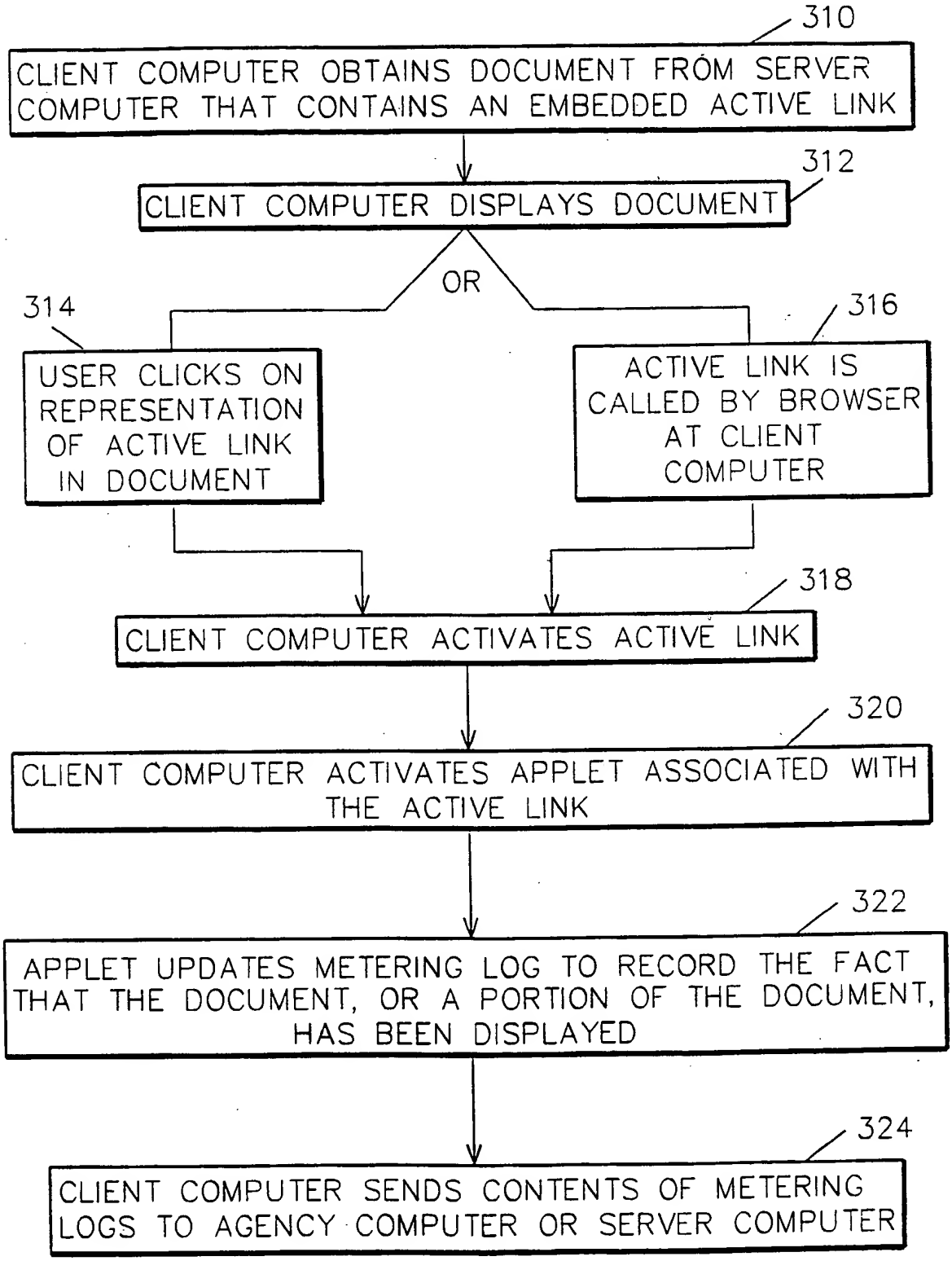
310
CLIENT COMPUTER OBTAINS DOCUMENT FROM SERVER COMPUTER THAT CONTAINS AN EMBEDDED ACTIVE LINK
312
CLIENT COMPUTER DISPLAYS DOCUMENT
OR
314
USER CLICKS ON REPRESENTATION OF ACTIVE LINK IN DOCUMENT
316
ACTIVE LINK IS CALLED BY BROWSER AT CLIENT COMPUTER
318
CLIENT COMPUTER ACTIVATES ACTIVE LINK
320
CLIENT COMPUTER ACTIVATES APPLET ASSOCIATED WITH THE ACTIVE LINK
322
APPLET UPDATES METERING LOG TO RECORD THE FACT THAT THE DOCUMENT, OR A PORTION OF THE DOCUMENT, HAS BEEN DISPLAYED
324
CLIENT COMPUTER SENDS CONTENTS OF METERING LOGS TO AGENCY COMPUTER OR SERVER COMPUTER

FIG. 8